

Everything you wanted to know about Data Analysis and Fitting but were afraid to ask

Peter Young

(Dated: October 16, 2012)

These notes discuss, in a style intended for physicists, how to average data and fit it to some functional form. I try to make clear what is being calculated, what assumptions are being made, and to give a derivation of results rather than just quote them. The aim is put a lot useful pedagogical material together in a convenient place. This manuscript is a substantial enlargement of lecture notes I prepared for the Bad Honnef School on “Efficient Algorithms in Computational Physics”, September 10–14, 2012.

Contents

I. Introduction	2
II. Averages and error bars	3
A. Basic Analysis	3
B. Advanced Analysis	9
1. Traditional method	10
2. Jackknife	12
3. Bootstrap	15
4. Jackknife or Bootstrap?	18
III. Fitting data to a model	18
A. Fitting to a straight line	20
B. Fitting to a polynomial	22
C. Error Bars	22
D. Fitting to a non-linear model	27
E. Confidence limits	28
F. Confidence limits by resampling the data	32
G. A tale of two probabilities. When can one rule out a fit?	33
A. Central Limit Theorem	35

B. The number of degrees of freedom	39
C. The chi-squared distribution and the goodness of fit parameter Q	40
D. Asymptotic standard error and how to get correct error bars from gnuplot	41
E. The distribution of fitted parameters determined from simulated datasets	43
F. The distribution of fitted parameters from repeated sets of measurements	44
G. Scripts for some data analysis and fitting tasks	46
1. Scripts for a jackknife analysis	46
a. Perl	46
b. Python	47
2. Scripts for a straight-line fit	48
a. Perl, writing out the formulae by hand	48
b. Python, writing out the formulae by hand	49
c. Python, using a built-in routine from scipy	50
d. Gnuplot	51
3. Scripts for a fit to a non-linear model	52
a. Python	53
b. Gnuplot	54
Acknowledgments	55
References	55

I. INTRODUCTION

These notes describe how to average and fit numerical data that you have obtained, presumably by some simulation.

Typically you will generate a set of values $x_i, y_i, \dots, i = 1, \dots, N$, where N is the number of measurements. The first thing you will want to do is to estimate various average values, and determine *error bars* on those estimates. As we shall see, this is straightforward if one wants to compute a single average, e.g. $\langle x \rangle$, but not quite so easy for more complicated averages such

as fluctuations in a quantity, $\langle x^2 \rangle - \langle x \rangle^2$, or combinations of measured values such as $\langle y \rangle / \langle x \rangle^2$. Averaging of data will be discussed in Sec. II.

Having obtained several good data points with error bars, you might want to fit this data to some model. Techniques for fitting data will be described in the second part of these notes in Sec. III

I find that the books on these topics usually fall into one of two camps. At one extreme, the books for physicists don't discuss all that is needed and rarely *prove* the results that they quote. At the other extreme, the books for mathematicians presumably prove everything but are written in a style of lemmas, proofs, ϵ 's and δ 's, and unfamiliar notation, which is intimidating to physicists. One exception, which finds a good middle ground, is Numerical Recipes [1] and the discussion of fitting given here is certainly influenced by Chap. 15 of that book. In these notes I aim to be fairly complete and also to derive the results I use, while the style is that of a physicist writing for physicists. I also include scripts in python, perl, and gnuplot to perform certain tasks in data analysis and fitting. For these reasons, these notes are perhaps rather lengthy. Nonetheless, I hope, that they will provide a useful reference.

II. AVERAGES AND ERROR BARS

A. Basic Analysis

Suppose we have a set of data from a simulation, x_i , ($i = 1, \dots, N$), which we shall refer to as a *sample* of data. This data will have some random noise so the x_i are not all equal. Rather they are governed by a distribution $P(x)$, *which we don't know*.

The distribution is normalized,

$$\int_{-\infty}^{\infty} P(x) dx = 1, \quad (1)$$

and is usefully characterized by its moments, where the n -th moment is defined by

$$\langle x^n \rangle = \int_{-\infty}^{\infty} x^n P(x) dx. \quad (2)$$

We will denote the average *over the exact distribution* by angular brackets. Of particular interest are the first and second moments from which one forms the mean μ and variance σ^2 , by

$$\mu \equiv \langle x \rangle \quad (3a)$$

$$\sigma^2 \equiv \langle (x - \langle x \rangle)^2 \rangle = \langle x^2 \rangle - \langle x \rangle^2. \quad (3b)$$

The term “standard deviation” is used for σ , the square root of the variance.

In this section we will estimate the mean $\langle x \rangle$, and the uncertainty in our estimate, from the N data points x_i . The determination of more complicated averages and resulting error bars will be discussed in Sec. II B

In order to obtain error bars we need to assume that the data are uncorrelated with each other. This is a crucial assumption, without which it is very difficult to proceed. However, it is not always clear if the data points are truly independent of each other; some correlations may be present but not immediately obvious. Here, we take the usual approach of assuming that even if there are some correlations, they are sufficiently weak so as not to significantly perturb the results of the analysis. In Monte Carlo simulations, measurements which differ by a sufficiently large number of Monte Carlo sweeps will be uncorrelated. More precisely the difference in sweep numbers should be greater than a “relaxation time”. This is exploited in the “binning” method in which the data used in the analysis is not the individual measurements, but rather an average over measurements during a range of Monte Carlo sweeps, called a “bin”. If the bin size is greater than the relaxation time, results from adjacent bins will be (almost) uncorrelated. A pedagogical treatment of binning has been given by Ambegaokar and Troyer [2]. Alternatively, one can do independent Monte Carlo runs, requilibrating each time, and use, as individual data in the analysis, the average from each run.

The information *from the data* is usefully encoded in two parameters, the sample mean \bar{x} and the sample standard deviation s which are defined by¹

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i, \quad (4a)$$

$$s^2 = \frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2. \quad (4b)$$

In statistics, notation is often confusing but crucial to understand. Here, an average indicated by an over-bar, $\overline{\cdots}$, is an average over the *sample of N data points*. This is to be distinguished from an exact average over the distribution $\langle \cdots \rangle$, as in Eqs. (3a) and (3b). The latter is, however, just a theoretical construct since we *don’t know* the distribution $P(x)$, only the set of N data points x_i

¹ The factor of $N-1$ rather than N in the expression for the sample variance in Eq. (4b) needs a couple of comments. Firstly, the final answer for the error bar on the mean, Eq. (16) below, will be independent of how this intermediate quantity is defined. Secondly, the N terms in Eq. (4b) are not all independent since \bar{x} , which is itself given by the x_i , is subtracted. Rather, as will be discussed more in the section on fitting, Sec. III, there are really only $N-1$ independent variables (called the “number of degrees of freedom” in the fitting context) and so dividing by $N-1$ rather than N has a rational basis. However, this is not essential and many authors divide by N in their definition of the sample variance.

which have been sampled from it.

Next we derive two simple results which will be useful later:

1. The mean of the sum of N independent variables *with the same distribution* is N times the mean of a single variable, and
2. The variance of the sum of N independent variables *with the same distribution* is N times the variance of a single variable.

The result for the mean is obvious since, defining $X = \sum_{i=1}^N x_i$,

$$\langle X \rangle = \sum_{i=1}^N \langle x_i \rangle = N \langle x_i \rangle \quad \boxed{= N\mu.} \quad (5)$$

The result for the standard deviation needs a little more work:

$$\sigma_X^2 \equiv \langle X^2 \rangle - \langle X \rangle^2 \quad (6a)$$

$$= \sum_{i,j=1}^N (\langle x_i x_j \rangle - \langle x_i \rangle \langle x_j \rangle) \quad (6b)$$

$$= \sum_{i=1}^N (\langle x_i^2 \rangle - \langle x_i \rangle^2) \quad (6c)$$

$$= N (\langle x^2 \rangle - \langle x \rangle^2) \quad (6d)$$

$$\boxed{= N\sigma^2.} \quad (6e)$$

To get from Eq. (6b) to Eq. (6c) we note that, for $i \neq j$, $\langle x_i x_j \rangle = \langle x_i \rangle \langle x_j \rangle$ since x_i and x_j are assumed to be statistically independent. (This is where the statistical independence of the data is needed.)

If the means and standard deviations are not all the same, then the above results generalize to

$$\langle X \rangle = \sum_{i=1}^N \mu_i, \quad (7a)$$

$$\langle \sigma_X^2 \rangle = \sum_{i=1}^N \sigma_i^2. \quad (7b)$$

Now we describe an important thought experiment. Let's *suppose* that we could repeat the set of N measurements *very many* many times, each time obtaining a value of the sample average \bar{x} . From these results we could construct a distribution, $\tilde{P}(\bar{x})$, for the sample average as shown in Fig. 1.

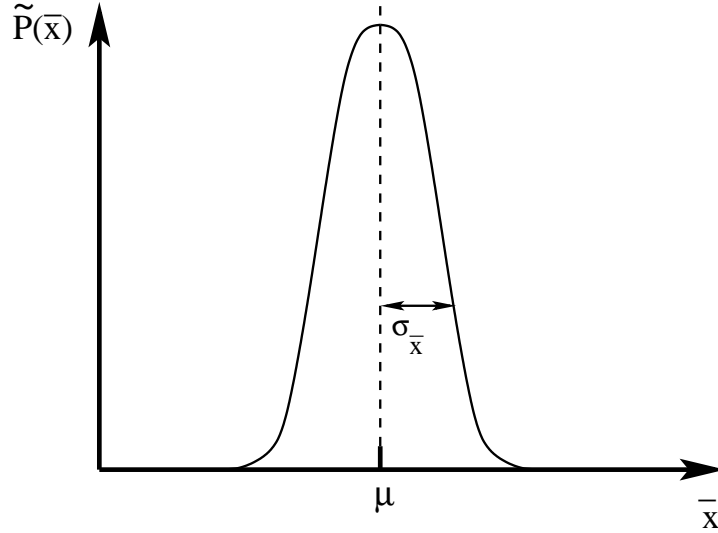


FIG. 1: The distribution of results for the sample mean \bar{x} obtained by repeating the measurements of the N data points x_i many times. The average of this distribution is μ , the exact average value of x . The mean, \bar{x} , obtained from one sample of data typically differs from μ by an amount of order $\sigma_{\bar{x}}$, the standard deviation of the distribution $\tilde{P}(\bar{x})$.

If we do enough repetitions we are effectively averaging over the exact distribution. Hence the average of the sample mean, \bar{x} , over very many repetitions of the data, is given by

$$\langle \bar{x} \rangle = \frac{1}{N} \sum_{i=1}^N \langle x_i \rangle = \langle x \rangle \equiv \mu, \quad (8)$$

i.e. it is the exact average over the distribution of x , as one would intuitively expect, see Fig. 1. Eq. (8) also follows from Eq. (5) by noting that $\bar{x} = X/N$.

In fact, though, we have only the *one* set of data, so we can not determine μ exactly. However, Eq. (8) shows that

$$\boxed{\text{the best estimate of } \mu \text{ is } \bar{x},} \quad (9)$$

i.e. the sample mean, since averaging the sample mean over many repetitions of the N data points gives the true mean of the distribution, μ . An estimate like this, which gives the exact result if averaged over many repetitions of the experiment, is said to be unbiased.

We would also like an estimate of the uncertainty, or “error bar”, in our estimate of \bar{x} for the exact average μ . We take $\sigma_{\bar{x}}$, the standard deviation in \bar{x} (obtained if one did many repetitions of

the N measurements), to be the uncertainty, or error bar, in \bar{x} . The reason is that $\sigma_{\bar{x}}$ is the width of the distribution $\tilde{P}(\bar{x})$, shown in Fig. 1, so a *single* estimate \bar{x} typically differs from the exact result μ by an amount of this order. The variance $\sigma_{\bar{x}}^2$ is given by

$$\sigma_{\bar{x}}^2 \equiv \langle \bar{x}^2 \rangle - \langle \bar{x} \rangle^2 = \frac{\sigma^2}{N}, \quad (10)$$

which follows from Eq. (6e) since $\bar{x} = X/N$.

The problem with Eq. (10) is that **we don't know** σ^2 since it is a function of the exact distribution $P(x)$. We do, however, know the *sample* variance s^2 , see Eq. (4b), and the average of this over many repetitions of the N data points, is equal to σ^2 since

$$\langle s^2 \rangle = \frac{1}{N-1} \sum_{i=1}^N \langle x_i^2 \rangle - \frac{1}{N(N-1)} \sum_{i=1}^N \sum_{j=1}^N \langle x_i x_j \rangle \quad (11a)$$

$$= \frac{N}{N-1} \langle x^2 \rangle - \frac{1}{N(N-1)} [N(N-1) \langle x \rangle^2 + N \langle x^2 \rangle] \quad (11b)$$

$$= [\langle x^2 \rangle - \langle x \rangle^2] \quad (11c)$$

$$= \sigma^2. \quad (11d)$$

To get from Eq. (11a) to Eq. (11b), we have separated the terms with $i = j$ in the last term of Eq. (11a) from those with $i \neq j$, and used the fact that each of the x_i is chosen from the same distribution and is statistically independent of the others. It follows from Eq. (11d) that

$$\boxed{\text{the best estimate of } \sigma^2 \text{ is } s^2,} \quad (12)$$

since averaging s^2 over many repetitions of N data points gives σ^2 . The estimate for σ^2 in Eq. (12) is therefore unbiased.

Combining Eqs. (10) and (12) gives

$$\boxed{\text{the best estimate of } \sigma_{\bar{x}}^2 \text{ is } \frac{s^2}{N},} \quad (13)$$

since this estimate is also unbiased. We have now obtained, using only information from the data, that the mean is given by

$$\boxed{\mu = \bar{x} \pm \sigma_{\bar{x}},} \quad (14)$$

where

$$\boxed{\sigma_{\bar{x}} = \frac{s}{\sqrt{N}},} \quad (15)$$

which we can write explicitly in terms of the data points as

$$\sigma_{\bar{x}} = \left[\frac{1}{N(N-1)} \sum_{i=1}^N (x_i - \bar{x})^2 \right]^{1/2}. \quad (16)$$

Remember that \bar{x} and s are the mean and standard deviation of the (one set) of data that is available to us, see Eqs. (4a) and (4b).

As an example, suppose $N = 5$ and the data points are

$$x_i = 10, 11, 12, 13, 14, \quad (17)$$

(not very random looking data it must be admitted!). Then, from Eq. (4a) we have $\bar{x} = 12$, and from Eq. (4b)

$$s^2 = \frac{1}{4} [(-2)^2 + (-1)^2 + 0^2 + 1^2 + 2^2] = \frac{5}{2}. \quad (18)$$

Hence, from Eq. (15),

$$\sigma_{\bar{x}} = \frac{1}{\sqrt{5}} \sqrt{\frac{5}{2}} = \frac{1}{\sqrt{2}}. \quad (19)$$

so

$$\mu = \bar{x} \pm \sigma_{\bar{x}} = 12 \pm \frac{1}{\sqrt{2}}. \quad (20)$$

How does the error bar decrease with the number of statistically independent data points N ? Equation (11d) states that the expectation value of s^2 is equal to σ^2 and hence, from Eq. (15), we see that

the error bar in the mean goes down like $1/\sqrt{N}$.

Hence, to reduce the error bar by a factor of 10 one needs 100 times as much data. This is discouraging, but is a fact of life when dealing with random noise.

For Eq. (15) to be really useful we need to know the probability that the true answer μ lies more than $\sigma_{\bar{x}}$ away from our estimate \bar{x} . Fortunately, for large N , the central limit theorem, derived in Appendix A, tells us (for distributions where the first two moments are finite) that the distribution of \bar{x} is a Gaussian. For this distribution we know that the probability of finding a result more than one standard deviation away from the mean is 32%, more than two standard deviations is 4.5% and more than three standard deviations is 0.3%. Hence we expect that most of the time \bar{x} will be within $\sigma_{\bar{x}}$ of the correct result μ , and only occasionally will be more than two times $\sigma_{\bar{x}}$ from

it. Even if N is not very large, so there are some deviations from the Gaussian form, the above numbers are often a reasonable guide.

However, as emphasized in appendix A, distributions which occur in nature typically have much more weight in the tails than a Gaussian. As a result, the weight in the tails of the distribution of *the sum* can also be much larger than for a Gaussian even for quite large values of N , see Fig. 6. It follows that the probability of an “outlier” can be much higher than that predicted for a Gaussian distribution, as anyone who has invested in the stock market knows well!

B. Advanced Analysis

In Sec. II A we learned how to estimate a simple average, such as $\mu_x \equiv \langle x \rangle$, plus the error bar in that quantity, from a set of data x_i . Trivially this method also applies to a *linear* combination of different averages, μ_x, μ_y, \dots etc. However, we often need more complicated, *non-linear* functions of averages. One example is the fluctuations in a quantity, i.e. $\langle x^2 \rangle - \langle x \rangle^2$. Another example is a dimensionless combination of moments, which gives information about the *shape* of a distribution independent of its overall scale. Such quantities are very popular in finite-size scaling (FSS) analyses since the FSS form is simpler than for quantities with dimension. An popular example, first proposed by Binder, is $\langle x^4 \rangle / \langle x^2 \rangle^2$, which is known as the “kurtosis” (frequently a factor of 3 is subtracted to make it zero for a Gaussian).

Hence, in this section we consider how to determine *non-linear functions* of averages of one or more variables, $f(\mu_y, \mu_z, \dots)$, where

$$\mu_y \equiv \langle y \rangle, \quad (21)$$

etc. For example, the two quantities mentioned in the previous paragraph correspond to

$$f(\mu_y, \mu_z) = \mu_y - \mu_z^2, \quad (22)$$

with $y = x^2$ and $z = x$ and

$$f(\mu_y, \mu_z) = \frac{\mu_y}{\mu_z^2}, \quad (23)$$

with $y = x^4$ and $z = x^2$.

The natural estimate of $f(\mu_y, \mu_z)$ from the sample data is clearly $f(\bar{y}, \bar{z})$. However, it will take some more thought to estimate the error bar in this quantity. The traditional way of doing this is called “error propagation”, described in Sec. IIB 1 below. However, it is now more common to

use either “jackknife” or “bootstrap” procedures, described in Secs. [IIB 2](#) and [IIB 3](#). At the price of some additional computation, which is no difficulty when done on a modern computer (though it would have been tedious in the old days when statistics calculations were done by hand), these methods automate the calculation of the error bar.

Furthermore, the estimate of $f(\mu_y, \mu_z)$ turns out to have some *bias* if f is a non-linear function. Usually this is small effect because it is order $1/N$, see for example Eq. [\(30\)](#) below, whereas the statistical error is of order $1/\sqrt{N}$. Since N is usually large, the bias is generally much less than the statistical error and so can generally be neglected. In any case, the jackknife and bootstrap methods also enable one to eliminate the leading ($\sim 1/N$) contribution to the bias in a automatic fashion.

1. Traditional method

First we will discuss the traditional method, known as error propagation, to compute the error bar and bias. We expand $f(\bar{y}, \bar{z})$ about $f(\mu_y, \mu_z)$ up to second order in the deviations:

$$f(\bar{y}, \bar{z}) = f(\mu_y, \mu_z) + (\partial_{\mu_y} f) \delta_{\bar{y}} + (\partial_{\mu_z} f) \delta_{\bar{z}} + \frac{1}{2} (\partial_{\mu_y \mu_y}^2 f) \delta_{\bar{y}}^2 + (\partial_{\mu_y \mu_z}^2 f) \delta_{\bar{y}} \delta_{\bar{z}} + \frac{1}{2} (\partial_{\mu_z \mu_z}^2 f) \delta_{\bar{z}}^2 + \dots, \quad (24)$$

where

$$\delta_{\bar{y}} = \bar{y} - \mu_y, \quad (25)$$

etc.

The terms of first order in the δ 's in Eq. [\(24\)](#) give the leading contribution to the error, but would average to zero if the procedure were to be repeated many times. However, the terms of second order do not average to zero and so give the leading contribution to the bias. We now estimate that bias.

Averaging Eq. [\(24\)](#) over many repetitions, and noting that

$$\langle \delta_{\bar{y}}^2 \rangle = \langle \bar{y}^2 \rangle - \langle \bar{y} \rangle^2 \equiv \sigma_{\bar{y}}^2, \quad \langle \delta_{\bar{z}}^2 \rangle = \langle \bar{z}^2 \rangle - \langle \bar{z} \rangle^2 \equiv \sigma_{\bar{z}}^2, \quad \langle \delta_{\bar{y}} \delta_{\bar{z}} \rangle = \langle \bar{y} \bar{z} \rangle - \langle \bar{y} \rangle \langle \bar{z} \rangle \equiv \sigma_{\bar{y} \bar{z}}^2, \quad (26)$$

we get

$$\langle f(\bar{y}, \bar{z}) \rangle - f(\mu_y, \mu_z) = \frac{1}{2} (\partial_{\mu_y \mu_y}^2 f) \sigma_{\bar{y}}^2 + (\partial_{\mu_y \mu_z}^2 f) \sigma_{\bar{y} \bar{z}}^2 + \frac{1}{2} (\partial_{\mu_z \mu_z}^2 f) \sigma_{\bar{z}}^2. \quad (27)$$

As shown in Eq. [\(13\)](#) our estimate of $\sigma_{\bar{y}}^2$ is N^{-1} times the sample variance (which we now call s_{yy}^2), and similarly for $\sigma_{\bar{z}}^2$. In the same way, our estimate of $\sigma_{\bar{y} \bar{z}}^2$ is N^{-1} times the sample *covariance* of

y and z , defined by

$$s_{yz}^2 = \frac{1}{N-1} \sum_{i=1}^N (y_i - \bar{y}) (z_i - \bar{z}) . \quad (28)$$

Hence, from Eq. (27), we have

$$f(\mu_y, \mu_z) = \langle f(\bar{y}, \bar{z}) \rangle - \frac{1}{N} \left[\frac{1}{2} (\partial_{\mu_y \mu_y}^2 f) s_{yy}^2 + (\partial_{\mu_y \mu_z}^2 f) s_{yz}^2 + \frac{1}{2} (\partial_{\mu_z \mu_z}^2 f) s_{zz}^2 \right] , \quad (29)$$

where the leading contribution to the bias is given by the $1/N$ term. Note that the bias term is “self-averaging”, i.e. the fluctuations in it are small relative to the average (by a factor of $1/\sqrt{N}$) when averaging over many repetitions of the data. It follows from Eq. (29) that if one wants to eliminate the leading contribution to the bias one should

$$\boxed{\text{estimate } f(\mu_y, \mu_z) \text{ from } f(\bar{y}, \bar{z}) - \frac{1}{N} \left[\frac{1}{2} (\partial_{\mu_y \mu_y}^2 f) s_{yy}^2 + (\partial_{\mu_y \mu_z}^2 f) s_{yz}^2 + \frac{1}{2} (\partial_{\mu_z \mu_z}^2 f) s_{zz}^2 \right]} . \quad (30)$$

As claimed earlier, the bias correction is of order $1/N$. Note that it vanishes if f is a linear function, as shown in Sec. II A. The generalization to functions of more than two averages, $f(\mu_y, \mu_z, \mu_w, \dots)$, is obvious.

Next we discuss the leading *error* in using $f(\bar{y}, \bar{z})$ as an estimate for $f(\mu_y, \mu_z)$. This comes from the terms linear in the δ 's in Eq. (24). Just including these terms we have

$$\langle f(\bar{y}, \bar{z}) \rangle = f(\mu_y, \mu_z) , \quad (31a)$$

$$\langle f^2(\bar{y}, \bar{z}) \rangle = f^2(\mu_y, \mu_z) + (\partial_{\mu_y} f)^2 \langle \delta_{\bar{y}}^2 \rangle + 2(\partial_{\mu_y} f) (\partial_{\mu_z} f) \langle \delta_{\bar{y}} \delta_{\bar{z}} \rangle + (\partial_{\mu_z} f)^2 \langle \delta_{\bar{z}}^2 \rangle . \quad (31b)$$

Hence

$$\begin{aligned} \sigma_f^2 &\equiv \langle f^2(\bar{y}, \bar{z}) \rangle - \langle f(\bar{y}, \bar{z}) \rangle^2 \\ &= (\partial_{\mu_y} f)^2 \langle \delta_{\bar{y}}^2 \rangle + 2(\partial_{\mu_y} f) (\partial_{\mu_z} f) \langle \delta_{\bar{y}} \delta_{\bar{z}} \rangle + (\partial_{\mu_z} f)^2 \langle \delta_{\bar{z}}^2 \rangle . \end{aligned} \quad (32)$$

As above, we use s_{yy}^2/N as an estimate of $\langle \delta_{\bar{y}}^2 \rangle$ and similarly for the other terms. Hence

$$\boxed{\text{the best estimate of } \sigma_f^2 \text{ is } \frac{1}{N} (\partial_{\mu_y} f)^2 s_{yy}^2 + 2(\partial_{\mu_y} f) (\partial_{\mu_z} f) s_{yz}^2 + (\partial_{\mu_z} f)^2 s_{zz}^2} . \quad (33)$$

This estimate is unbiased to leading order in N . Note that we need to keep track not only of fluctuations in y and z , characterized by their variances s_{yy}^2 and s_{zz}^2 , but also cross correlations between y and z , characterized by their covariance s_{yz}^2 .

Hence, still to leading order in N , we get

$$\boxed{f(\mu_y, \mu_z) = f(\bar{y}, \bar{z}) \pm \sigma_f} , \quad (34)$$

where we estimate the error bar σ_f from Eq. (33) which shows that it is of order $1/\sqrt{N}$. Again, the generalization to functions of more than two averages is obvious.

Note that in the simple case studied in Sec. II A where there is only one set of variables x_i and $f = \mu_x$, Eq. (29) tells us that there is no bias, which is correct, and Eq. (33) gives an expression for the error bar which agrees with Eq. (15).

In Eqs. (30) and (33) we need to keep track how errors in the individual quantities like \bar{y} propagate to the estimate of the function f . This requires inputting by hand the various partial derivatives into the analysis program, and keeping track of all the variances and covariances. In the next two sections we see how *resampling* the data automatically takes account of error propagation without needing to input the partial derivatives and keep track of variances and covariances. These approaches, known as jackknife and bootstrap, provide a *fully automatic* method of determining error bars and bias.

2. Jackknife

We define the i -th jackknife estimate, y_i^J ($i = 1, 2, \dots, N$) to be the average over all data in the sample *except the point i* , i.e.

$$y_i^J \equiv \frac{1}{N-1} \sum_{j \neq i} y_j. \quad (35)$$

We also define corresponding jackknife estimates of the function f (again for concreteness we will assume that f is a function of just 2 averages but the generalization will be obvious):

$$f_i^J \equiv f(y_i^J, z_i^J). \quad (36)$$

In other words, we use the jackknife values, y_i^J, z_i^J , rather than the sample means, \bar{y}, \bar{z} , as the arguments of f . For example a jackknife estimate of the Binder ratio $\langle x^4 \rangle / \langle x^2 \rangle^2$ is

$$f_i^J = \frac{(N-1)^{-1} \sum_{j, (j \neq i)} x_j^4}{\left[(N-1)^{-1} \sum_{j \neq i} x_j^2 \right]^2} \quad (37)$$

The overall jackknife estimate of $f(\mu_y, \mu_z)$ is then the average over the N jackknife estimates f_i^J :

$$\boxed{\overline{f^J} \equiv \frac{1}{N} \sum_{i=1}^N f_i^J.} \quad (38)$$

It is straightforward to show that if f is a linear function of μ_y and μ_z then $\overline{f^J} = f(\bar{y}, \bar{z})$, i.e. the jackknife and standard averages are identical. However, when f is not a linear function, so there

is bias, there *is* a difference, and we will now show the resampling carried out in the jackknife method can be used to determine bias and error bars in an automated way.

We proceed as for the derivation of Eq. (29), which we now write as

$$f(\mu_y, \mu_z) = \langle f(\bar{y}, \bar{z}) \rangle - \frac{A}{N} - \frac{B}{N^2} + \dots, \quad (39)$$

where A is the term in rectangular brackets in Eq. (29), and we have added the next order correction. The jackknife data sets have $N - 1$ points with the same distribution as the N points in the actual distribution, and so the bias in the jackknife average will be of the same form, with the same values of A and B , but with N replaced by $N - 1$, i.e.

$$f(\mu_y, \mu_z) = \langle \bar{f}^J \rangle - \frac{A}{N - 1} - \frac{B}{(N - 1)^2} \dots. \quad (40)$$

We can therefore eliminate the leading contribution to the bias by forming an appropriate linear combination of $f(\bar{y}, \bar{z})$ and \bar{f}^J , namely

$$f(\mu_y, \mu_z) = N \langle f(\bar{y}, \bar{z}) \rangle - (N - 1) \langle \bar{f}^J \rangle + O\left(\frac{1}{N^2}\right). \quad (41)$$

It follows that, to eliminate the leading bias without computing partial derivatives, one should

$$\boxed{\text{estimate } f(\mu_y, \mu_z) \text{ from } N f(\bar{y}, \bar{z}) - (N - 1) \bar{f}^J.} \quad (42)$$

The bias is then of order $1/N^2$. However, as mentioned earlier, bias is usually not a big problem because, even without eliminating the leading contribution, the bias is of order $1/N$ whereas the statistical error is of order $1/\sqrt{N}$ which is much bigger if N is large. In most cases, therefore, N is sufficiently large that one can use *either* the usual average $f(\bar{y}, \bar{z})$, or the jackknife average \bar{f}^J in Eq. (38), to estimate $f(\mu_y, \mu_z)$, since the difference between them will be much smaller than the statistical error. In other words, elimination of the leading bias using Eq. (42) is usually not necessary.

Next we show that the jackknife method gives error bars, which agree with Eq. (33) but without the need to explicitly keep track of the partial derivatives and the variances and covariances.

We define the variance of the jackknife averages by

$$\sigma_{f^J}^2 \equiv \overline{(f^J)^2} - \left(\bar{f}^J\right)^2, \quad (43)$$

where

$$\overline{(f^J)^2} = \frac{1}{N} \sum_{i=1}^N (f_i^J)^2. \quad (44)$$

Using Eqs. (36) and (38), we expand $\overline{f^J}$ away from the exact result $f(\mu_y, \mu_z)$. Just including the leading contribution gives

$$\begin{aligned}\overline{f^J} - f(\mu_y, \mu_z) &= \frac{1}{N} \sum_{i=1}^N [(\partial_{\mu_y} f)(y_i^J - \mu_y) + (\partial_{\mu_z} f)(z_i^J - \mu_z)] \\ &= \frac{1}{N(N-1)} \sum_{i=1}^N [(\partial_{\mu_y} f) \{N(\overline{y} - \mu_y) - (y_i - \mu_y)\} + (\partial_{\mu_z} f) \{N(\overline{z} - \mu_z) - (z_i - \mu_z)\}] \\ &= (\partial_{\mu_y} f)(\overline{y} - \mu_y) + (\partial_{\mu_z} f)(\overline{z} - \mu_z).\end{aligned}\tag{45}$$

Similarly we find

$$\begin{aligned}\overline{(f^J)^2} &= \frac{1}{N} \sum_{i=1}^N [f(\mu_y, \mu_z) + (\partial_{\mu_y} f)(y_i^J - \mu_y) + (\partial_{\mu_z} f)(z_i^J - \mu_z)]^2 \\ &= f^2(\mu_y, \mu_z) + 2f(\mu_y, \mu_z) [(\partial_{\mu_y} f)(\overline{y} - \mu_y) + (\partial_{\mu_z} f)(\overline{z} - \mu_z)] \\ &\quad + (\partial_{\mu_y} f)^2 \left[(\overline{y} - \mu_y)^2 + \frac{s_{yy}^2}{N(N-1)} \right] + (\partial_{\mu_z} f)^2 \left[(\overline{z} - \mu_z)^2 + \frac{s_{zz}^2}{N(N-1)} \right] \\ &\quad + 2(\partial_{\mu_y} f)(\partial_{\mu_z} f) \left[(\overline{y} - \mu_y)(\overline{z} - \mu_z) + \frac{s_{yz}^2}{N(N-1)} \right].\end{aligned}\tag{46}$$

Hence, from Eqs. (43) and (45), the variance in the jackknife estimates is given by

$$\sigma_{f^J}^2 = \frac{1}{N(N-1)} [(\partial_{\mu_y} f)^2 s_{yy}^2 + (\partial_{\mu_z} f)^2 s_{zz}^2 + 2(\partial_{\mu_y} f)(\partial_{\mu_z} f) s_{yz}] ,\tag{47}$$

which is just $1/(N-1)$ times σ_f^2 , the estimate of the square of the error bar in $f(\overline{y}, \overline{z})$ given in Eq. (33). Hence

the jackknife estimate for σ_f is $\sqrt{N-1} \sigma_{f^J}$.

(48)

Note that this is directly obtained from the jackknife estimates without having to put in the partial derivatives by hand. Note too that the $\sqrt{N-1}$ factor is in the *numerator* whereas the factor of \sqrt{N} in Eq. (15) is in the *denominator*. Intuitively the reason for this difference is that the jackknife estimates are very close since they would all be equal except that each one omits just one data point.

If N is very large, roundoff errors could become significant from having to subtract large, almost equal, numbers to get the error bar from the jackknife method. It is then advisable to group the N data points into N_{group} groups (or “bins”) of data and take, as individual data points in the jackknife analysis, the average of the data in each group. The above results clearly go through with N replaced by N_{group} .

To summarize this subsection, to estimate $f(\mu_y, \mu_z)$ one can use either $f(\bar{y}, \bar{z})$ or the jackknife average $\overline{f^J}$ in Eq. (38). The error bar in this estimate, σ_f , is related to the standard deviation in the jackknife estimates σ_{f^J} by Eq. (48).

3. Bootstrap

The bootstrap, like the jackknife, is a resampling of the N data points. Whereas jackknife considers N new data sets, each of containing all the original data points minus one, bootstrap uses N_{boot} data sets each containing N points obtained by random (Monte Carlo) sampling of the original set of N points. During the Monte Carlo sampling, the probability that a data point is picked is $1/N$ irrespective of whether it has been picked before. (In the statistics literature this is called picking from a set “with replacement”.) Hence a given data point x_i will, *on average*, appear once in each Monte Carlo-generated data set, but may appear not at all, or twice, and so on. The probability that x_i appears n_i times is close to a Poisson distribution with mean unity. However, it is not exactly Poissonian because of the constraint in Eq. (49) below. It turns out that we shall need to include the deviation from the Poisson distribution even for large N . We shall use the term “bootstrap” data sets to denote the Monte Carlo-generated data sets.

More precisely, let us suppose that the number of times x_i appears in a bootstrap data set is n_i . Since each bootstrap dataset contains exactly N data points, we have the constraint

$$\sum_{i=1}^N n_i = N. \quad (49)$$

Consider one of the N variables x_i . Each time we generate an element in a bootstrap dataset the probability that it is x_i is $1/N$, which we will denote by p . From standard probability theory, the probability that x_i occurs n_i times is given by a binomial distribution

$$P(n_i) = \frac{N!}{n_i! (N - n_i)!} p^{n_i} (1 - p)^{N - n_i}. \quad (50)$$

The mean and standard deviation of a binomial distribution are given by

$$[n_i]_{\text{MC}} = Np = 1, \quad (51)$$

$$[n_i^2]_{\text{MC}} - [n_i]_{\text{MC}}^2 = Np(1 - p) = 1 - \frac{1}{N}, \quad (52)$$

where $[\dots]_{\text{MC}}$ denotes an exact average over bootstrap samples (for a fixed original data set x_i). For $N \rightarrow \infty$, the binomial distribution goes over to a Poisson distribution, for which the factor of $1/N$ in Eq. (52) does not appear. We assume that N_{boot} is sufficiently large that the bootstrap

average we carry out reproduces this result with sufficient accuracy. Later, we will discuss what values for N_{boot} are sufficient in practice. Because of the constraint in Eq. (49), n_i and n_j (with $i \neq j$) are not independent and we find, by squaring Eq. (49) and using Eqs. (51) and (52), that

$$[n_i n_j]_{\text{MC}} - [n_i]_{\text{MC}} [n_j]_{\text{MC}} = -\frac{1}{N} \quad (i \neq j). \quad (53)$$

First of all we just consider the simple average $\mu_x \equiv \langle x \rangle$, for which, of course, the standard methods in Sec. II A suffice, so bootstrap is not necessary. However, this will show how to get averages and error bars in a simple case, which we will then generalize to non-linear functions of averages.

We denote the average of x for a given bootstrap data set by x_α^B , where α runs from 1 to N_{boot} , *i.e.*

$$x_\alpha^B = \frac{1}{N} \sum_{i=1}^N n_i^\alpha x_i. \quad (54)$$

We then compute the bootstrap average of the mean of x and the bootstrap variance in the mean, by averaging over all the bootstrap data sets. We assume that N_{boot} is large enough for the bootstrap average to be exact, so we can use Eqs. (52) and (53). The result is

$$\overline{x^B} \equiv \frac{1}{N_{\text{boot}}} \sum_{\alpha=1}^{N_{\text{boot}}} x_\alpha^B = \frac{1}{N} \sum_{i=1}^N [n_i]_{\text{MC}} x_i = \frac{1}{N} \sum_{i=1}^N x_i = \bar{x} \quad (55)$$

$$\sigma_{x^B}^2 \equiv \overline{(x^B)^2} - (\overline{x^B})^2 = \frac{1}{N^2} \left(1 - \frac{1}{N}\right) \sum_i x_i^2 - \frac{1}{N^3} \sum_{i \neq j} x_i x_j, \quad (56)$$

where

$$\overline{(x^B)^2} \equiv \frac{1}{N_{\text{boot}}} \sum_{\alpha=1}^{N_{\text{boot}}} \left[(x_\alpha^B)^2 \right]_{\text{MC}}. \quad (57)$$

We now average Eqs. (55) and (56) over many repetitions of the original data set x_i . Averaging Eq. (55) gives

$$\langle \overline{x^B} \rangle = \langle \bar{x} \rangle = \langle x \rangle \equiv \mu_x. \quad (58)$$

This shows that the bootstrap average $\overline{x^B}$ is an unbiased estimate of the exact average μ_x . Averaging Eq. (56) gives

$$\langle \sigma_{x^B}^2 \rangle = \frac{N-1}{N^2} \sigma^2 = \frac{N-1}{N} \sigma_{\bar{x}}^2, \quad (59)$$

where we used Eq. (10) to get the last expression. Since $\sigma_{\bar{x}}$ is the uncertainty in the sample mean, we see that

the bootstrap estimate of $\sigma_{\bar{x}}$ is $\sqrt{\frac{N}{N-1}} \sigma_{x^B}$.

(60)

Remember that σ_{x^B} is the standard deviation of the bootstrap data sets. Usually N is sufficiently large that the square root in Eq. (60) can be replaced by unity.

As for the jackknife, these results can be generalized to finding the error bar in some possibly non-linear function, $f(\mu_y, \mu_z)$, rather than for μ_x . One computes the bootstrap estimates for $f(\mu_y, \mu_z)$, which are

$$f_\alpha^B = f(y_\alpha^B, z_\alpha^B). \quad (61)$$

In other words, we use the bootstrap values, y_α^B, z_α^B , rather than the sample means, \bar{y}, \bar{z} , as the arguments of f . The final bootstrap estimate for $f(\mu_y, \mu_z)$ is the average of these, *i.e.*

$$\overline{f^B} = \frac{1}{N_{\text{boot}}} \sum_{\alpha=1}^{N_{\text{boot}}} f_\alpha^B. \quad (62)$$

Following the same methods in the jackknife section, one obtains the error bar, σ_f , in $f(\mu_y, \mu_z)$. The result is

$$\text{the bootstrap estimate for } \sigma_f \text{ is } \sqrt{\frac{N}{N-1}} \sigma_{f^B}, \quad (63)$$

where

$$\sigma_{f^B}^2 = \overline{(f^B)^2} - (\overline{f^B})^2, \quad (64)$$

is the variance of the bootstrap estimates. Here

$$\overline{(f^B)^2} \equiv \frac{1}{N_{\text{boot}}} \sum_{\alpha=1}^{N_{\text{boot}}} (f_\alpha^B)^2. \quad (65)$$

Usually N is large enough that the factor of $\sqrt{N/(N-1)}$ in Eq. (63) can be replaced by unity. Equation (63) corresponds to the result Eq. (60) which we derived for the special case of $f = \mu_x$.

Again, following the same path as in the jackknife section, it is straightforward to show that the bias of the estimates in Eqs. (62) and (63) is of order $1/N$ and so vanishes for $N \rightarrow \infty$. However, if N is not too large it may be useful to eliminate the leading contribution to the bias in the mean, as we did for jackknife in Eq. (42). The result is that one should

$$\text{estimate } f(\mu_y, \mu_z) \text{ from } 2f(\bar{y}, \bar{z}) - \overline{f^B}. \quad (66)$$

The bias in Eq. (66) is of order $1/N^2$, whereas $f(\bar{y}, \bar{z})$ and $\overline{f^B}$ each have a bias of order $1/N$. However, it is not normally necessary to eliminate the bias since, if N is large, the bias is much smaller than the statistical error.

I have not systematically studied the values of N_{boot} that are needed in practice to get accurate estimates for the error. It seems that N_{boot} in the range 100 to 500 is typically chosen, and this seems to be adequate irrespective of how large N is.

To summarize this subsection, to estimate $f(\mu_y, \mu_z)$ one can either use $f(\bar{y}, \bar{z})$, or the bootstrap average in Eq. (62), and the error bar in this estimate, σ_f , is related to the standard deviation in the bootstrap estimates by Eq. (63).

4. Jackknife or Bootstrap?

The jackknife approach involves less calculation than bootstrap, and is fine for estimating combinations of moments of the measured quantities. Furthermore, identical results are obtained each time jackknife is run on the same set of data, which is not the case for bootstrap. However, the range of the jackknife estimates is very much smaller, by a factor of \sqrt{N} for large N , than the scatter of averages which would be obtained from individual data sets, see Eq. (48). By contrast, for bootstrap, σ_{fB} , which measures the deviation of the bootstrap estimates f_α^B from the result for the single actual data set $f(\bar{y}, \bar{z})$, is equal to σ_f , the deviation of the average of a single data set from the exact result $f(\mu_y, \mu_z)$ (if we replace the factor of $N/(N-1)$ by unity, see Eq. (63)). This is the main strength of the bootstrap approach; it samples the full range of the distribution of the sample distribution. Hence, if you want to generate data which covers the full range then should use bootstrap. This is useful in fitting, see for example, Sec. III F. However, if you just want to generate error bars on combinations of moments quickly and easily, then use jackknife.

III. FITTING DATA TO A MODEL

A good reference for the material in this section is Chapter 15 of Numerical Recipes [1].

Frequently we are given a set of data points $(x_i, y_i), i = 1, 2, \dots, N$, with corresponding error bars, σ_i , through which we would like to fit to a smooth function $f(x)$. The function could be straight line (the simplest case), a higher order polynomial, or a more complicated function. The fitting function will depend on M “fitting parameters”, a_α and we would like the “best” fit obtained by adjusting these parameters. We emphasize that a fitting procedure should not only

1. give the values of the fit parameters, but also
2. provide error estimates on those parameters, and

3. provide a measure of how good the fit is.

If the result of part 3 is that the fit is very poor, the results of parts 1 and 2 are probably meaningless.

The definition of “best” is not unique. However, the most useful choice, and the one nearly always taken, is “least squares”, in which one minimizes the sum of the squares of the difference between the observed y -value, y_i , and the fitting function evaluated at x_i , weighted appropriately by the error bars since if some points have smaller error bars than others the fit should be closer to those points. The quantity to be minimized, called “chi-squared”,² and written mathematically as χ^2 , is therefore

$$\chi^2 = \sum_{i=1}^N \left(\frac{y_i - f(x_i)}{\sigma_i} \right)^2. \quad (67)$$

Often we assume that the distribution of the errors is Gaussian, since, according to the central limit theorem discussed in Appendix A, the sum of N independent random variables has a Gaussian distribution (under fairly general conditions) if N is large. However, distributions which occur in nature usually have more weight in the “tails” than a Gaussian, and as a result, even for moderately large values of N , the probability of an “outlier” might be much bigger than expected from a Gaussian, see Fig. 6.

If the errors *are* distributed with a Gaussian distribution, and if $f(x)$ has the *exact* values of the fit parameters, then χ^2 in Eq. (67) is a sum of squares of N random variables with a Gaussian distribution with mean zero and standard deviation unity. However, when we have minimized the value of χ^2 with respect to the M fitting parameters a_α the terms are not all independent. It turns out, see Appendix B, that, at least for a linear model (which we define below), the distribution of χ^2 at the minimum is that of the sum of the squares of $N - M$ (not N) Gaussian random variable with zero mean and standard deviation unity³. We call $N - M$ the “number of degrees of freedom” (N_{DOF}). The χ^2 distribution is discussed in Appendix C. The formula for it is Eq. (C6).

The simplest problems are where the fitting function is a *linear function of the parameters*. We shall call this a *linear model*. Examples are a straight line ($M = 2$),

$$y = a_0 + a_1 x, \quad (68)$$

² χ^2 should be thought of as a single variable rather than the square of something called χ . This notation is standard.

³ Although this result is only valid if the fitting model is linear in the parameters, it is usually taken to be a reasonable approximation for non-linear models as well.

and an m -th order polynomial ($M = m + 1$),

$$y = a_0 + a_1x + a_2x^2 + \cdots + a_mx^m = \sum_{\alpha=0}^m a_\alpha x^\alpha, \quad (69)$$

where the parameters to be adjusted are the a_α . (Note that we are *not* stating here that y has to be a linear function of x , only of the fit parameters a_α .)

An example where the fitting function depends *non*-linearly on the parameters is

$$y = a_0x^{a_1} + a_2. \quad (70)$$

Linear models are fairly simply because, as we shall see, the parameters are determined by *linear* equations, which, in general, have a unique solution that can be found by straightforward methods. However, for fitting functions which are non-linear functions of the parameters, the resulting equations are *non-linear* which may have many solutions or none at all, and so are much less straightforward to solve. We shall discuss fitting to both linear and non-linear models in these notes.

Sometimes a non-linear model can be transformed into a linear model by a change of variables. For example, if we want to fit to

$$y = a_0x^{a_1}, \quad (71)$$

which has a non-linear dependence on a_1 , taking logs gives

$$\ln y = \ln a_0 + a_1 \ln x, \quad (72)$$

which is a *linear* function of the parameters $a'_0 = \ln a_0$ and a_1 . Fitting a straight line to a log-log plot is a very common procedure in science and engineering. However, it should be noted that transforming the data does not exactly take Gaussian errors into Gaussian errors, though the difference will be small if the errors are “sufficiently small”. For the above log transformation this means $\sigma_i/y_i \ll 1$, i.e. the *relative* error is much less than unity.

A. Fitting to a straight line

To see how least squares fitting works, consider the simplest case of a straight line fit, Eq. (68), for which we have to minimize

$$\chi^2(a_0, a_1) = \sum_{i=1}^N \left(\frac{y_i - a_0 - a_1x_i}{\sigma_i} \right)^2, \quad (73)$$

with respect to a_0 and a_1 . Differentiating χ^2 with respect to these parameters and setting the results to zero gives

$$a_0 \sum_{i=1}^N \frac{1}{\sigma_i^2} + a_1 \sum_{i=1}^N \frac{x_i}{\sigma_i^2} = \sum_{i=1}^N \frac{y_i}{\sigma_i^2}, \quad (74a)$$

$$a_0 \sum_{i=1}^N \frac{x_i}{\sigma_i^2} + a_1 \sum_{i=1}^N \frac{x_i^2}{\sigma_i^2} = \sum_{i=1}^N \frac{x_i y_i}{\sigma_i^2}. \quad (74b)$$

We write this as

$$U_{00} a_0 + U_{01} a_1 = v_0, \quad (75a)$$

$$U_{10} a_0 + U_{11} a_1 = v_1, \quad (75b)$$

where

$$U_{\alpha\beta} = \sum_{i=1}^N \frac{x_i^{\alpha+\beta}}{\sigma_i^2}, \quad \text{and} \quad (76)$$

$$v_\alpha = \sum_{i=1}^N \frac{y_i x_i^\alpha}{\sigma_i^2}. \quad (77)$$

The matrix notation, while an overkill here, will be convenient later when we do a general polynomial fit. Note that $U_{10} = U_{01}$. (More generally, later on, U will be a symmetric matrix). Equations (75) are two linear equations in two unknowns. These can be solved by eliminating one variable, which immediately gives an equation for the second one. The solution can also be determined from

$$a_\alpha = \sum_{\beta=0}^m (U^{-1})_{\alpha\beta} v_\beta, \quad (78)$$

(where we have temporarily generalized to a polynomial of order m). For the straight-line fit, the inverse of the 2×2 matrix U is given, according to standard rules, by

$$U^{-1} = \frac{1}{\Delta} \begin{pmatrix} U_{11} & -U_{01} \\ -U_{01} & U_{00} \end{pmatrix} \quad (79)$$

where

$$\Delta = U_{00}U_{11} - U_{01}^2, \quad (80)$$

and we have noted that U is symmetric so $U_{01} = U_{10}$. The solution for a_0 and a_1 is therefore given by

$$a_0 = \frac{U_{11} v_0 - U_{01} v_1}{\Delta}, \quad (81a)$$

$$a_1 = \frac{-U_{01} v_0 + U_{00} v_1}{\Delta}. \quad (81b)$$

We see that it is straightforward to determine the slope, a_1 , and the intercept, a_0 , of the fit from Eqs. (76), (77), (80) and (81) using the N data points (x_i, y_i) , and their error bars σ_i .

B. Fitting to a polynomial

Frequently we need to fit to a higher order polynomial than a straight line, in which case we minimize

$$\chi^2(a_0, a_1, \dots, a_m) = \sum_{i=1}^N \left(\frac{y_i - \sum_{\alpha=0}^m a_\alpha x_i^\alpha}{\sigma_i} \right)^2 \quad (82)$$

with respect to the $(m+1)$ parameters a_α . Setting to zero the derivatives of χ^2 with respect to the a_α gives

$$\boxed{\sum_{\beta=0}^m U_{\alpha\beta} a_\beta = v_\alpha}, \quad (83)$$

where $U_{\alpha\beta}$ and v_α have been defined in Eqs. (76) and (77). Eq. (83) represents $M = m+1$ *linear* equations, one for each value of α . Their solution is again given by Eq. (78), i.e. it is expressed in terms of the inverse matrix U^{-1} .

C. Error Bars

In addition to the best fit values of the parameters we also need to determine the error bars in those values. Interestingly, this information is *also* contained in the matrix U^{-1} .

First of all, we explain the significance of error bars in fit parameters. We assume that the data is described by a model with a particular set of parameters \vec{a}^{true} which, unfortunately, we don't know. If we were, somehow, to have many real data sets each one would give a different set of fit parameters $\vec{a}^{(i)}, i = 0, 1, 2, \dots$, because of noise in the data, *clustered about the true set* \vec{a}^{true} . Projecting on to a single fit parameter, a_1 say, there will be a distribution of values $P(a_1)$ centered on a_1^{true} with standard deviation σ_1 , see the top part of Fig. 2. Typically the value of a_1 obtained from our *one actual data set*, $a_1^{(0)}$, will lie within about σ_1 of a_1 . Hence we define the error bar to be σ_1 .

Unfortunately, we can't determine the error bar this way because we have only one actual data set, which we denote here by $y_i^{(0)}$ to distinguish it from other data sets that we will introduce. Our

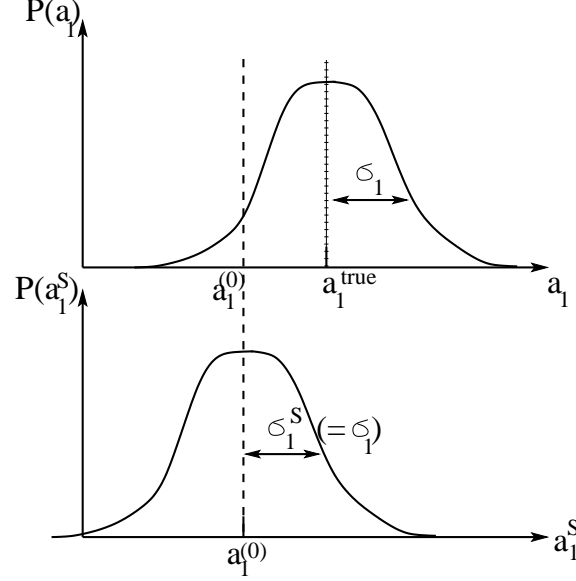


FIG. 2: The top figure shows the distribution of one of the fit parameters a_1 if one could obtain many real data sets. The distribution has standard deviation σ_1 about the true value a_1^{true} and is Gaussian if the noise on the data is Gaussian. In fact, however, we have only one actual data set which has fit parameter $a_1^{(0)}$, and this typically lies within about σ_1 of a_1^{true} . Hence we define the error bar on the estimate of a_1^{true} to be σ_1 . However, we cannot calculate σ_1 directly from the distribution of a_1 because we have only one the one value, $a_1^{(0)}$. However, we can generate many *simulated* data sets from the one actual set and hence we can estimate the standard deviation, σ_1^S , of the distribution of the resulting fit parameter a_1^S , which is shown in the lower figure. This distribution is centered about the value from the actual data, $a_1^{(0)}$, and has standard deviation, σ_1^S . The important point is that if one assumes a linear model then one can show that $\sigma_1^S = \sigma_1$, see text. Even if the model is non linear, one usually assumes that the difference in the standard deviations is sufficiently small that one can still equate the true error bar with the standard deviation from the simulated data sets. We emphasize that the error bar quoted by fitting programs is actually σ_1^S , and this is assumed to equal σ_1 . Furthermore, as shown in Appendices E and F, if the noise on the data is Gaussian (and the model is linear) both the distributions in this figure are also Gaussian.

actual data set gives one set of fit parameters, which we call $\vec{a}^{(0)}$. Suppose, however, we were to generate many *simulated* data sets from of the one which is available to us, by generating random values (possibly with a Gaussian distribution though this won't be necessary yet) centered at the y_i with standard deviation σ_i . Fitting each simulated dataset would give different values for \vec{a} , *clustered now about $\vec{a}^{(0)}$* , see the bottom part of Fig. (2). We now come to an important, but rarely discussed, point:

We assume that the standard deviation of the fit parameters of these simulated data sets about $\vec{a}^{(0)}$, which we will be able to calculate from the single set of data available

to us, is equal to the standard deviation of the fit parameters of real data sets \vec{a} about \vec{a}^{true} . The latter is what we *really* want to know (since it is our estimate of the error bar on \vec{a}^{true}) but can't determine directly. See Fig. 2 for an illustration. In fact we show in the text below that this assumption is correct for a linear model (and for a non-linear model if the range of parameter values is small enough that it can be represented by an effective linear model). Even if the model is non linear, one usually assumes that the two standard deviations are sufficiently close that the difference is not important. Furthermore, we show in Appendices E and F that if the noise on the data is Gaussian (and the model is linear), the two distributions in Fig. (2) are also both Gaussian.

Hence, as stated above, to derive the error bars in the fit parameters we take simulated values of the data points, y_i^S , which vary by some amount δy_i^S about $y_i^{(0)}$, i.e. $\delta y_i^S = y_i^S - y_i^{(0)}$, with a standard deviation given by the error bar σ_i . The fit parameters of this simulated data set, \vec{a}^S , then deviate from $\vec{a}^{(0)}$ by an amount $\delta \vec{a}^S$ where

$$\delta a_\alpha^S = \sum_{i=1}^N \frac{\partial a_\alpha}{\partial y_i} \delta y_i^S. \quad (84)$$

Averaging over fluctuations in the y_i^S we get the variance of a_α^S to be

$$(\sigma_\alpha^S)^2 \equiv \langle (\delta a_\alpha^S)^2 \rangle = \sum_{i=1}^N \sigma_i^2 \left(\frac{\partial a_\alpha}{\partial y_i} \right)^2, \quad (85)$$

since $\langle (\delta y_i^S)^2 \rangle = \sigma_i^2$, and the data points y_i are statistically independent. Writing Eq. (78) explicitly in terms of the data values,

$$a_\alpha = \sum_{\beta} (U^{-1})_{\alpha\beta} \sum_{i=1}^N \frac{y_i x_i^\beta}{\sigma_i^2}, \quad (86)$$

and noting that U is independent of the y_i , we get

$$\frac{\partial a_\alpha}{\partial y_i} = \sum_{\beta} (U^{-1})_{\alpha\beta} \frac{x_i^\beta}{\sigma_i^2}. \quad (87)$$

Substituting into Eq. (85) gives

$$(\sigma_\alpha^S)^2 = \sum_{\beta, \gamma} (U^{-1})_{\alpha\beta} (U^{-1})_{\alpha\gamma} \left[\sum_{i=1}^N \frac{x_i^{\beta+\gamma}}{\sigma_i^2} \right]. \quad (88)$$

The term in rectangular brackets is just $U_{\beta\gamma}$, and so, noting that U is given by Eq. (76) and is symmetric, the last equation reduces to

$$(\sigma_\alpha^S)^2 = (U^{-1})_{\alpha\alpha}. \quad (89)$$

Recall that σ_α^S is the standard deviation of the fitted parameter values about the $\vec{a}^{(0)}$ when constructing simulated data sets from the one set of data that is available to us. However, the error bar is defined to be the standard deviation the fitted parameter values would have relative to a_α^{true} if we could average over many actual data sets. To determine this quantity we simply repeat the above calculation with $\delta y_i = y_i - y_i^{\text{true}}$ in which y_i is the value of the i -th data point in one of the actual data sets. The result is identical to Eq. (89), namely

$$\boxed{\sigma_\alpha^2 = (U^{-1})_{\alpha\alpha}}, \quad (90)$$

in which U is the *same* in Eq. (90) as in Eq. (89) because U is a constant, for a linear model, independent of the y_i or the fit parameters a_α . Hence σ_α in Eq. (90) is the error bar in a_α .

In addition to error bars, we also need a parameter to describe the quality of the fit. A useful quantity is the probability that, given the fit, the data could have occurred with a χ^2 greater than or equal to the value found. This is generally denoted by Q and is given by Eq. (C9) assuming the data have Gaussian noise. Note that the effects of *non-Gaussian* statistics is to increase the probability of outliers, so fits with a fairly small value of Q , say around 0.01, may be considered acceptable. However, fits with a *very* small value of Q should not be trusted and the values of the fit parameters are probably meaningless in these cases.

For the case of a straight line fit, the inverse of U is given explicitly in Eq. (79). Using this information, and the values of (x_i, y_i, σ_i) for the data in Fig. 3, the fit parameters (assuming a straight line fit) are

$$a_0 = 0.84 \pm 0.32, \quad (91)$$

$$a_1 = 2.05 \pm 0.11, \quad (92)$$

in which the error bars on the fit parameters on a_0 and a_1 , which are denoted by σ_0 and σ_1 , are determined from Eq. (90). The data was generated by starting with $y = 1 + 2x$ and then adding some noise with zero mean. Hence the fit should be consistent with $y = 1 + 2x$ within the error bars, and it is. The value of χ^2 is 7.44 so $\chi^2/N_{\text{DOF}} = 7.44/9 = 0.866$ and the quality of fit parameter, given by Eq. (C9), is $Q = 0.592$ which is good.

We call U^{-1} the “*covariance matrix*”. Its off-diagonal elements are also useful since they contain information about correlations between the fitted parameters. More precisely, one can show, following the lines of the above derivation of σ_α^2 , that the correlation of fit parameters α and

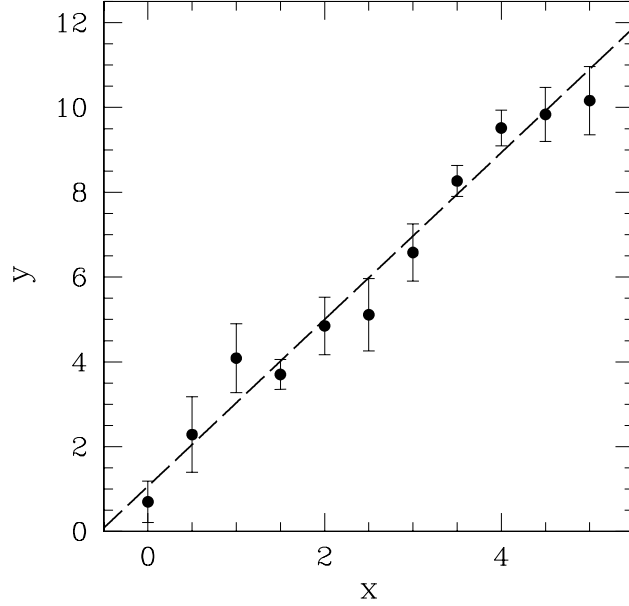


FIG. 3: An example of a straight-line fit to a set of data with error bars.

β , known mathematically as their “covariance”, is given by the appropriate off-diagonal element of the covariance matrix,

$$\text{Cov}(\alpha, \beta) \equiv \langle \delta a_\alpha \delta a_\beta \rangle = (U^{-1})_{\alpha\beta} . \quad (93)$$

The correlation coefficient, $r_{\alpha\beta}$, which is a dimensionless measure of the correlation between δa_α and δa_β lying between -1 and 1 , is given by

$$r_{\alpha\beta} = \frac{\text{Cov}(\alpha, \beta)}{\sigma_\alpha \sigma_\beta} . \quad (94)$$

A good fitting program should output the correlation coefficients as well as the fit parameters, their error bars, the value of χ^2/N_{DOF} , and the goodness of fit parameter Q .

For a linear model, χ^2 is a quadratic function of the fit parameters and so the elements of the “*curvature matrix*”⁴, $(1/2) \partial^2 \chi^2 / \partial a_\alpha \partial a_\beta$ are constants, independent of the values of the fit parameters. In fact, we see from Eqs. (76) and (82) that

$$\frac{1}{2} \frac{\partial^2 \chi^2}{\partial a_\alpha \partial a_\beta} = U_{\alpha\beta} , \quad (95)$$

⁴ It is conventional to include the factor of $1/2$.

so the curvature matrix is equal to U , given by Eq. (76) for a polynomial fit.

If we fit to a *general* linear model, writing

$$f(x) = \sum_{\alpha=1}^M a_{\alpha} X_{\alpha}(x), \quad (96)$$

where $X_1(x), X_2(x), \dots, X_M(x)$ a fixed functions of x called basis functions, the curvature matrix is given by

$$U_{\alpha\beta} = \sum_{i=1}^N \frac{X_{\alpha}(x_i) X_{\beta}(x_i)}{\sigma_i^2}. \quad (97)$$

Similarly, the quantities v_{α} in Eq. (77) become

$$v_{\alpha} = \sum_{i=1}^N \frac{y_i X_{\alpha}(x_i)}{\sigma_i^2}, \quad (98)$$

for a general set of basis functions, and best fit parameters are given by the solution of the M linear equations

$$\sum_{\beta=1}^M U_{\alpha\beta} a_{\beta} = v_{\alpha}, \quad (99)$$

for $\alpha = 1, 2, \dots, M$. Note that for a linear model the curvature matrix U is a constant, independent of the fit parameters. However, U is not constant for a non-linear model.

D. Fitting to a non-linear model

As for linear models, one minimizes χ^2 in Eq. (67). The difference is that the resulting equations are non-linear so there might be many solutions or non at all. Techniques for solving the coupled non-linear equations invariably require specifying an initial value for the variables a_{α} . The most common method for fitting to non-linear models is the Levenberg-Marquardt (LM) method, see e.g. Numerical Recipes [1]. Implementing the Numerical Recipes code for LM is a little complicated because it requires the user to provide a routine for the derivatives of χ^2 with respect to the fit parameters as well as for χ^2 itself, and to check for convergence. Alternatively, one can use the fitting routines in the `scipy` package of `python` or use `gnuplot`. But see the comments in Appendix D about getting the error bars in the parameters correct. This applies when fitting to linear as well as non-linear models. Gnuplot and scipy scripts for fitting to a non-linear model are given in Appendix G.

One difference from fitting to a linear model is that the curvature matrix, defined by the LHS of Eq. (95), is not constant but is a function of the fit parameters. Hence it is no longer true that the standard deviations of the two distributions in Fig. 2 are equal. However, it still generally assumed that the difference is small enough to be unimportant and hence that the covariance matrix, which is now defined to be the inverse of the curvature matrix *at the minimum of χ^2* , still gives information about error bars on the fit parameters. This is discussed more in the next two subsections, in which we point out, however, that a more detailed analysis is needed if the model is non-linear and the spread of fitted parameters is sufficiently large that it cannot be represented by an effective linear model, i.e. χ^2 is not well fitted by a parabola over the needed range of parameter values.

As a reminder:

- The *curvature matrix* is defined in general by the LHS of Eq. (95), which, for a linear model, is equivalent to Eq. (97) (Eq. (76) for a polynomial fit.)
- The *covariance matrix* is the inverse of the curvature matrix at the minimum of χ^2 (the last remark being only needed for a non-linear model). Its diagonal elements give error bars on the fit parameters according to Eq. (90) (but see the caveat in the previous paragraph for non-linear models) and its off-diagonal elements give correlations between fit parameters according to Eqs. (93) and (94).

E. Confidence limits

In the last two subsections we showed that the diagonal elements of the covariance matrix give an error bar on the fit parameters. In this section we extend the notion of error bar to embrace the concept of a “confidence limit”.

There is a theorem [1] which states that, for a linear model, if we take simulated data sets assuming Gaussian noise in the data about the actual data points, and compute the fit parameters $\vec{a}^{S(i)}, i = 1, 2, \dots$ for each data set, then the probability distribution of the \vec{a}^S is given by the multi-variable Gaussian distribution

$$P(\vec{a}^S) \propto \exp \left(-\frac{1}{2} \sum_{\alpha, \beta} \delta a_\alpha^S U_{\alpha\beta} \delta a_\beta^S \right), \quad (100)$$

where $\delta \vec{a}^S \equiv \vec{a}^{S(i)} - \vec{a}^{(0)}$ and U , given by Eq. (97), is the curvature matrix which can also be defined in terms of the second derivative of χ^2 according to Eq. (95). A proof of this result is given in

Appendix E. It applies for a linear model with Gaussian noise, and also for a non-linear model if the uncertainties in the parameters do not extend outside a region where an effective linear model could be used. (In the latter case one still needs a non-linear routine to *find* the best parameters). Note that for a non-linear model, U is not a constant and is the curvature *at the minimum* of χ^2 .

From Eq. (95) the change in χ^2 as the parameters are varied away from the minimum is given by

$$\Delta\chi^2 \equiv \chi^2(\vec{a}^{S(i)}) - \chi^2(\vec{a}^{(0)}) = \sum_{\alpha,\beta} \delta a_\alpha^S U_{\alpha\beta} \delta a_\beta^S, \quad (101)$$

in which the χ^2 are all evaluated from the single (actual) data set $y_i^{(0)}$. Equation (100) can therefore be written as

$$P(\vec{a}^S) \propto \exp\left(-\frac{1}{2}\Delta\chi^2\right). \quad (102)$$

We remind the reader that we have assumed the noise in the data is Gaussian and that either the model is linear or, if non-linear, the uncertainties in the parameters do not extend outside a region where an effective linear model could be used.

Hence the probability of a particular deviation, $\delta\vec{a}^S$, of the fit parameters in a simulated data set away from the parameters in the *actual* data set, depends on how much this change increases χ^2 (evaluated from the actual data set) away from the minimum. In general a “confidence limit” is the range of fit parameter values such that $\Delta\chi^2$ is less than some specified value. The simplest case, and the only one we discuss here, is the variation of *one* variable at a time, though multi-variate confidence limits can also be defined, see Numerical Recipes [1].

We therefore consider the change in χ^2 when one variable, a_1^S say, is held at a specified value, and all the others ($\beta = 2, 3, \dots, M$) are varied in order to minimize χ^2 . Minimizing $\Delta\chi^2$ in Eq. (101) with respect to a_β^S gives

$$\sum_{\gamma=1}^M U_{\beta\gamma} \delta a_\gamma^S = 0, \quad (\beta = 2, 3, \dots, M). \quad (103)$$

The corresponding sum for $\beta = 1$, namely $\sum_{\gamma=1}^M U_{1\gamma} \delta a_\gamma^S$, is not zero because δa_1 is fixed. It will be some number, c say. Hence we can write

$$\sum_{\gamma=1}^M U_{\alpha\gamma} \delta a_\gamma^S = c_\alpha, \quad (\alpha = 1, 2, \dots, M), \quad (104)$$

where $c_1 = c$ and $c_\beta = 0$ ($\beta \neq 1$). The solution is

$$\delta a_\alpha^S = \sum_{\beta=1}^M (U^{-1})_{\alpha\beta} c_\beta. \quad (105)$$

For $\alpha = 1$ this gives

$$c = \delta a_1^S / (U^{-1})_{11} . \quad (106)$$

Substituting Eq. (105) into Eq. (101), and using Eq. (106) we find that $\Delta\chi^2$ is related to $(\delta a_1^S)^2$ by

$$\Delta\chi^2 = \frac{(\delta a_1^S)^2}{(U^{-1})_{11}} . \quad (107)$$

(Curiously, the coefficient of $(\delta a_1)^2$ is one over the 11 element of the inverse of U , rather than U_{11} which is how it appears in Eq. (101) in which the $\beta \neq 1$ parameters are free rather than adjusted to minimize χ^2 .)

From Eq. (102) we finally get

$$P(a_1^S) \propto \exp\left(-\frac{1}{2} \frac{(\delta a_1^S)^2}{\sigma_1^2}\right) , \quad (108)$$

where

$$\sigma_1^2 = (U^{-1})_{11} . \quad (109)$$

As shown in Appendices E and F, Eqs. (100), (102) and (108) also apply, under the same conditions (linear model and Gaussian noise on the data) to the probability for $\delta a_1 \equiv a_1^{\text{true}} - a_1^{(0)}$, where we remind the reader that $a_1^{(0)}$ is the fit parameter obtained from the actual data, and a_1^{true} is the exact value. In other words the probability of the true value is given by

$$\boxed{P(\vec{a}^{\text{true}}) \propto \exp\left(-\frac{1}{2} \Delta\chi^2\right)} , \quad (110)$$

where

$$\Delta\chi^2 \equiv \chi^2(\vec{a}^{\text{true}}) - \chi^2(\vec{a}^{(0)}) , \quad (111)$$

in which we remind the reader that both values of χ^2 are evaluated from the single set of data available to us, $y_i^{(0)}$. Projecting onto a single parameter, as above, gives

$$\boxed{P(a_1^{\text{true}}) \propto \exp\left(-\frac{1}{2} \frac{(\delta a_1)^2}{\sigma_1^2}\right)} , \quad (112)$$

so $\langle(\delta a_1)^2\rangle = \sigma_1^2 = (U^{-1})_{11}$, in agreement with what we found earlier in Eq. (90). We emphasize that Eqs. (110) and (112) assumes Gaussian noise on the data points, and either the model is linear or, if non-linear, that the range of uncertainty in the parameters is small enough that a description in terms of an effective linear model is satisfactory.

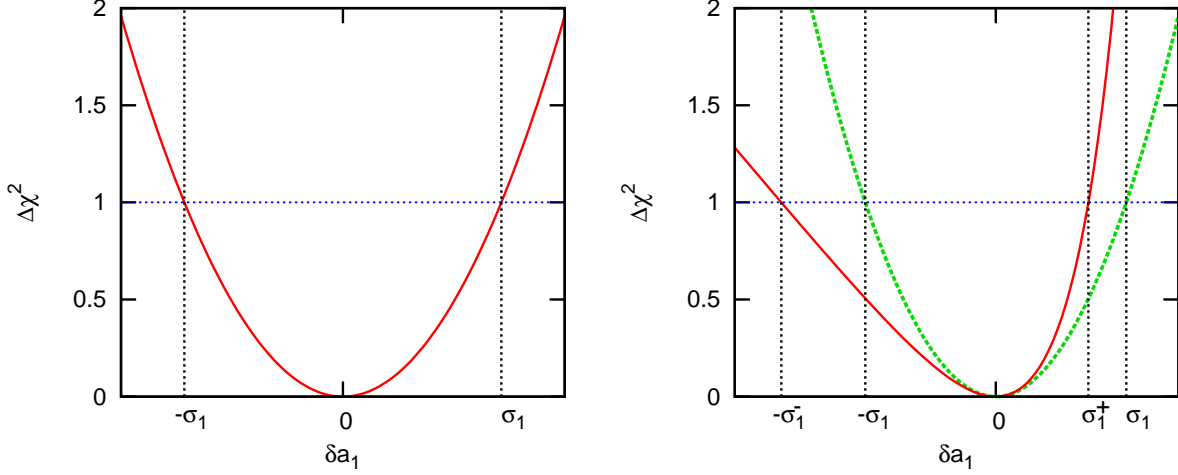


FIG. 4: **Left:** The change in χ^2 as a fit parameter a_1 is varied away from the value that minimizes χ^2 for a *linear* model. The shape is a parabola for which $\Delta\chi^2 = 1$ when $\delta a = \pm\sigma_1$, where σ_1 is the error bar.

Right: The solid curve is a sketch of the change in χ^2 for a *non-linear* model. The curve is no longer a parabola and is even non-symmetric. The dashed curve is a parabola which fits the solid curve at the minimum. The fitting program only has information about the *local* behavior at the minimum and so gives an error range $\pm\sigma_1$ where the value of the parabola is 1. However, the parameter a_1 is clearly more tightly constrained on the plus side than on the minus side, and a better way to determine the error range is to look *globally* and locate the values of δa_1 where $\Delta\chi^2 = 1$. This gives an error bar σ_1^+ on the plus side, and a different error bar, σ_1^- , on the minus side, both of which are different from σ_1 .

However we have done more than recover our earlier result, Eq. (90), by more complicated means since we have gained *additional* information. From the properties of a Gaussian distribution we now know that, from Eq. (112), the probability that a_α lies within one standard deviation σ_α of the value which minimizes χ^2 is 68%, the probability of its being within two standard deviations is 95.5%, and so on. Furthermore, from Eq. (110), we see that

if a single fit parameter is one standard deviation away from its value at the minimum of χ^2 (the other fit parameters being varied to minimize χ^2), then $\Delta\chi^2 = 1$.

This last sentence, and the corresponding equations Eqs. (110) and (112), are not valid for a non-linear model if the uncertainties of the parameters extends outside the range where an effective linear model can be used. In this situation, to get confidence limits, it is necessary to do a bootstrap resampling of the data, as discussed in the next subsection.

However, if one is not able to resample the data we argue that it is better to take the range where $\Delta\chi^2 \leq 1$ as an error bar for each parameter rather than the error bar determined from the curvature of χ^2 at the minimum, see Fig. 4. The left hand plot is for a linear model, for which the curve of $\Delta\chi^2$ against δa_1 is exactly a parabola, and the right hand plot is a sketch for a non-linear model, for which it is not a parabola though it has a quadratic variation about the minimum shown by the dashed curve. For the linear case, the values of δa_1 where $\Delta\chi^2 = 1$ are the *same* as the values $\pm\sigma_1$, where σ_1 is the standard error bar obtained from the *local* curvature in the vicinity of the minimum. However, for the non-linear case, the values of δa_1 where $\Delta\chi^2 = 1$ are *different* from $\pm\sigma_1$, and indeed the values on the positive and negative sides, σ_1^+ and σ_1^- , are not equal. For the data Fig. 4, it is clear that the value of a_1 is more tightly constrained on the positive side than the negative side, and so it is better to give the error bars as $+\sigma_1^+$ and $-\sigma_1^-$, obtained from the range where $\Delta\chi^2 \leq 1$, rather the symmetric range $\pm\sigma_1$. However, if possible, in these circumstances error bars and a confidence limit should actually be obtained from a bootstrap resampling of the data as discussed in the next section.

F. Confidence limits by resampling the data

More work is involved if one wants to get error bars and a confidence interval in the case where the model is non-linear and the range of parameter uncertainty extends outside the region where an effective linear model is adequate. Even for a linear model, we cannot convert $\Delta\chi^2$ into a confidence limit with a specific probability if the noise is non-Gaussian.

To proceed in these cases, one can bootstrap the individual data points as follows. Each data point (x_i, y_i) has error bar σ_i , which comes from averaging over N measurements, say. Generating bootstrap datasets by Monte Carlo sampling the N measurements, as discussed in Sec. II B 3, the distribution of the mean of each bootstrap dataset has a standard deviation equal to the estimate of standard deviation on the mean of the actual data set, see Eq. (63) (replacing the factor of $\sqrt{N/(N-1)}$ by unity which is valid since N is large in practice). Hence, if we generate N_{boot} bootstrap data sets, and fit each one, the scatter of the fitted parameter values will be a measure of the uncertainty in the values from the *actual* dataset. Forming a histogram of the values of a single parameter we can obtain a confidence interval within which 68%, say, of the bootstrap datasets lie (16% missing on either side) and interpret this range as a 68% confidence limit for the actual parameter value. The justification for this interpretation has been discussed in the statistics literature, see e.g. the references in Ref. [1], but I'm not able to go into the details here. Note

that this bootstrap approach could also be applied usefully for a *linear* model if the noise is not Gaussian.

Unfortunately, use of the bootstrap procedure to get error bars in fits to non-linear models does not yet seem to be a standard procedure in the statistical physics community.

Another possibility for a non-linear model, if one is confident that the noise is close to Gaussian, is to generate *simulated* data sets, assuming Gaussian noise on the y_i values with standard deviation given by the error bars σ_i . Each simulated dataset is fitted and the distribution of fitted parameters is determined. This corresponds to the analytical approach in Appendix E but without the assumption that the model can be represented by an effective linear one over of the needed parameter range.

G. A tale of two probabilities. When can one rule out a fit?

If the noise on the data is Gaussian, which we will assume throughout this subsection, we have, so far, considered two different probabilities.

Firstly, as discussed in Appendix C, the value of χ^2 is typically in the range $N_{\text{DOF}} \pm \sqrt{2N_{\text{DOF}}}$. The quality of fit parameter Q is the probability that, *given the fit*, the data could have this value of χ^2 or greater, and is given mathematically by Eq. (C9). It varies from unity when $\chi^2 \ll N_{\text{DOF}} - \sqrt{2N_{\text{DOF}}}$ to zero when $\chi^2 \gg N_{\text{DOF}} + \sqrt{2N_{\text{DOF}}}$. We emphasize that

Q is the probability of the data given the fit.

Secondly, in the context of error bars and confidence limits, we have discussed, in Eqs. (110) and (112), the probability that a fit parameter, a_1 say, takes a certain value relative to the optimal one. Equation (110) becomes very small when $\Delta\chi^2$ varies by much more than unity. Note that Eqs. (110) and (112) refer to the

relative probabilities of two fits, given the data.

At first, it seems curious that the probability Q remains significantly greater than zero if χ^2 changes by an amount of order $\sqrt{N_{\text{DOF}}}$, whereas if a fit parameter is changed by an amount such that χ^2 changes by of order $\sqrt{N_{\text{DOF}}}$, the probability of this value becomes extremely small, of order $\exp(-\text{const.} \sqrt{N_{\text{DOF}}})$, in this limit, see Eqs. (110). While there is no mathematical inconsistency, since the two probabilities refer to different situations (one is the probability of the data given the fit and the other is the relative probability of two fits given the data), it is useful to understand this difference intuitively.

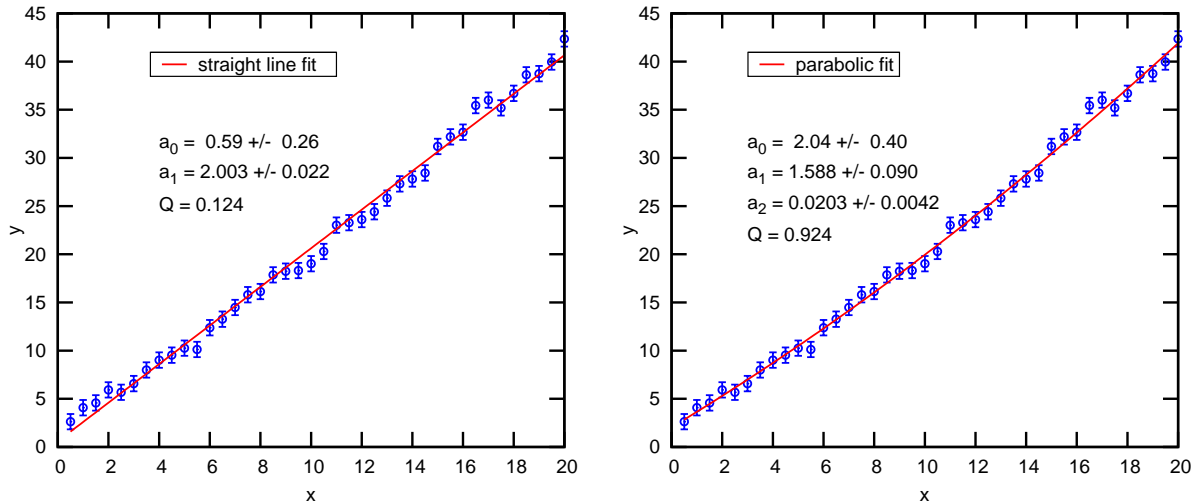


FIG. 5: **Left:** A straight-line fit to a data set. The value of Q is reasonable. However, one notices that the data is systematically above the fit for small x and for large x while it is below the fit for intermediate x . This is unlikely to happen by random chance. This remark is made more precise in the right figure.

Right: A parabolic fit to the same data set. The value of Q is larger than for the straight-line fit, but the main result is that the coefficient of the quadratic term is 5σ away from zero, showing that the straight-line fit in the left figure is much less likely than the parabolic fit.

We take, as an example, a problem where we want to know whether the data can be modeled by a straight line, or whether a quadratic term needs to be included as well. A set of data is shown in Fig. 5.

Looking at the left figure one sees that the data more or less agrees with the straight-line fit. However, one also sees systematic trends: the data is too high for small x and for high x , and too low for intermediate x . The probability that this trend would occur by chance is very low. Chi-squared just sums up the contributions from each data point and is insensitive to any systematic trend in the deviation of the data from the fit. Hence the value of χ^2 , in itself, does not tell us that this data is unlikely to be represented by a straight line. It is only when we add another parameter in the fit which corresponds to those correlations, that we realize the straight-line model is relatively very unlikely. In this case, the extra parameter is the coefficient of x^2 , and the resulting parabolic fit is shown in the right figure.

The qualitative comments in the last paragraph are made more precise by the parameters of the fits. The straight-line fit gives $a_0 = 0.59 \pm 0.26$, $a_1 = 2.003 \pm 0.022$ with $Q = 0.124$, whereas the parabolic fit gives $a_0 = 2.04 \pm 0.40$, $a_1 = 1.588 \pm 0.090$, $a_2 = 0.0203 \pm 0.0042$ with $Q = 0.924$. The actual parameters used to generate the data are $a_0 = 2$, $a_1 = 1.6$, $a_2 = 0.02$, and there is Gaussian

noise with standard deviation equal to 0.8. Although the quality of fit factor for the straight-line fit is reasonable, the quadratic fit strongly excludes having the fit parameter a_2 equal to zero, since zero is five standard deviations away from the best value. For a Gaussian distribution, the probability of a five-sigma deviation or greater is $\text{erfc}(5/\sqrt{2}) \simeq 6 \times 10^{-7}$. The difference in χ^2 for the quadratic fit, between the best fit and the fit forcing $a_2 = 0$, is $(0.0203/0.0042)^2 \simeq 23$ according to Eqs. (107) and (90). We conclude that, in this case, the straight-line model is unlikely to be correct.

The moral of this tale is that a reasonable value of Q does not, in itself, ensure that you have the right model. Another model might be very much more probable.

Appendix A: Central Limit Theorem

In this appendix we give a proof of the central limit theorem.

We assume a distribution that falls off sufficiently fast at $\pm\infty$ that the mean and variance are finite. This *excludes*, for example, the Lorentzian distribution:

$$P_{\text{Lor}} = \frac{1}{\pi} \frac{1}{1+x^2}. \quad (\text{A1})$$

A common distribution which *does* have a finite mean and variance is the Gaussian distribution,

$$P_{\text{Gauss}} = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]. \quad (\text{A2})$$

Using standard results for Gaussian integrals you should be able to show that the distribution is normalized and that the mean and standard deviation are μ and σ respectively.

Consider a distribution, *not necessarily Gaussian*, with a finite mean and distribution. We pick N random numbers x_i from such a distribution and form the sum

$$X = \sum_{i=1}^N x_i.$$

Note, we are assuming that all the random numbers have the *same* distribution.

The determination of the distribution of X , which we call $P_N(X)$, uses the Fourier transform of $P(x)$, called the “characteristic function” in the context of probability theory. This is defined by

$$Q(k) = \int_{-\infty}^{\infty} P(x) e^{ikx} dx.$$

Expanding out the exponential we can write $Q(k)$ in terms of the moments of $P(x)$

$$Q(k) = 1 + ik\langle x \rangle + \frac{(ik)^2}{2!} \langle x^2 \rangle + \frac{(ik)^3}{3!} \langle x^3 \rangle + \dots.$$

It will be convenient in what follows to write $Q(k)$ as an exponential, i.e.

$$\begin{aligned} Q(k) &= \exp \left[\ln \left(1 + ik\langle x \rangle + \frac{(ik)^2}{2!} \langle x^2 \rangle + \frac{(ik)^3}{3!} \langle x^3 \rangle + \dots \right) \right] \\ &= \exp \left[ik\mu - \frac{k^2\sigma^2}{2!} + \frac{c_3(ik)^3}{3!} + \frac{c_4(ik)^4}{4!} + \dots \right], \end{aligned} \quad (\text{A3})$$

where c_3 involves third and lower moments, c_4 involves fourth and lower moments, and so on. The c_n are called *cumulant* averages.

For the important case of a Gaussian, the Fourier transform is obtained by “completing the square”. The result is that the Fourier transform of a Gaussian is also a Gaussian, namely,

$$Q_{\text{Gauss}}(k) = \exp \left[ik\mu - \frac{k^2\sigma^2}{2} \right], \quad (\text{A4})$$

showing that the higher order cumulants, c_3, c_4 , etc. in Eq. (A3) *all vanish* for a Gaussian.

The distribution $P_N(x)$ can be expressed as

$$P_N(x) = \int_{-\infty}^{\infty} P(x_1) dx_1 \int_{-\infty}^{\infty} P(x_2) dx_2 \cdots \int_{-\infty}^{\infty} P(x_N) dx_N \delta(X - \sum_{i=1}^N x_i). \quad (\text{A5})$$

We evaluate this by using the integral representation of the delta function

$$\delta(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{ikx} dk, \quad (\text{A6})$$

so

$$P_N(X) = \int_{-\infty}^{\infty} \frac{dk}{2\pi} \int_{-\infty}^{\infty} P(x_1) dx_1 \int_{-\infty}^{\infty} P(x_2) dx_2 \cdots \int_{-\infty}^{\infty} P(x_N) dx_N \exp[ik(x_1 + x_2 + \cdots x_N - X)] \quad (\text{A7})$$

$$= \int_{-\infty}^{\infty} \frac{dk}{2\pi} Q(k)^N e^{-ikX}, \quad (\text{A8})$$

showing that the Fourier transform of $P_N(x)$, which we call $Q_N(k)$, is given by

$$Q_N(k) = Q(k)^N. \quad (\text{A9})$$

Consequently

$$Q_N(k) = \exp \left[ikN\mu - \frac{Nk^2\sigma^2}{2} + \frac{Nc_3(ik)^3}{4!} + \frac{Nc_4(ik)^4}{4!} + \dots \right]. \quad (\text{A10})$$

Comparing with Eq. (A3) we see that

the mean of the distribution of the sum of N independent and identically distributed random variables (the coefficient of $-ik$ in the exponential) is N times the mean of the distribution of one variable, and the variance of the distribution of the sum (the coefficient of $-k^2/2!$) is N times the variance of the distribution of one variable.

These are general statements applicable for *any* N and have already been derived in Sec. II A.

However, if N is *large* we can now go further. The distribution $P_N(X)$ is the inverse transform of $Q_N(k)$, see Eq. (A8), so

$$P_N(X) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \exp \left[-ikX' - \frac{Nk^2\sigma^2}{2!} + N \frac{c_3(ik)^3}{3!} + \frac{Nc_4(ik)^4}{4!} + \dots \right] dk, \quad (\text{A11})$$

where

$$X' = X - N\mu. \quad (\text{A12})$$

Looking at the $-Nk^2/2$ term in the exponential in Eq. (A11), we see that the integrand is significant for $k < k^*$, where $N\sigma^2(k^*)^2 = 1$, and negligibly small for $k \gg k^*$. However, for $0 < k < k^*$ the higher order terms in Eq. (A11), (i.e. those of order k^3, k^4 etc.) are very small since $N(k^*)^3 \sim N^{-1/2}$, $N(k^*)^4 \sim N^{-1}$ and so on. Hence the terms of higher order than k^2 in Eq. (A11), do not contribute for large N and so

$$\lim_{N \rightarrow \infty} P_N(X) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \exp \left[-ikX' - \frac{Nk^2\sigma^2}{2} \right] dk. \quad (\text{A13})$$

In other words, for large N the distribution is the Fourier transform of a Gaussian, which, as we know, is also a Gaussian. Completing the square in Eq. (A13) gives

$$\begin{aligned} \lim_{N \rightarrow \infty} P_N(X) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \exp \left[-\frac{N\sigma^2}{2} \left(k - \frac{iX'}{N\sigma^2} \right)^2 \right] dk \exp \left[-\frac{(X')^2}{2N\sigma^2} \right] \\ &= \boxed{\frac{1}{\sqrt{2\pi N} \sigma} \exp \left[-\frac{(X - N\mu)^2}{2N\sigma^2} \right]}, \end{aligned} \quad (\text{A14})$$

where, in the last line, we used Eq. (A12). This is a Gaussian with mean $N\mu$ and variance $N\sigma^2$. Equation (A14) is the central limit theorem in statistics. It tells us that,

for $N \rightarrow \infty$, the distribution of the sum of N independent and identically distributed random variables is a *Gaussian* whose mean is N times the mean, μ , of the distribution of one variable, and whose variance is N times the variance of the distribution of one variable, σ^2 , *independent of the form of the distribution of one variable, $P(x)$* , provided only that μ and σ are finite.

The central limit theorem is of such generality that it is extremely important. It is the reason why the Gaussian distribution has such a preeminent place in the theory of statistics.

Note that if the distribution of the individual x_i is Gaussian, then the distribution of the sum of N variables is *always* Gaussian, even for N small. This follows from Eq. (A9) and the fact that the Fourier transform of a Gaussian is a Gaussian.

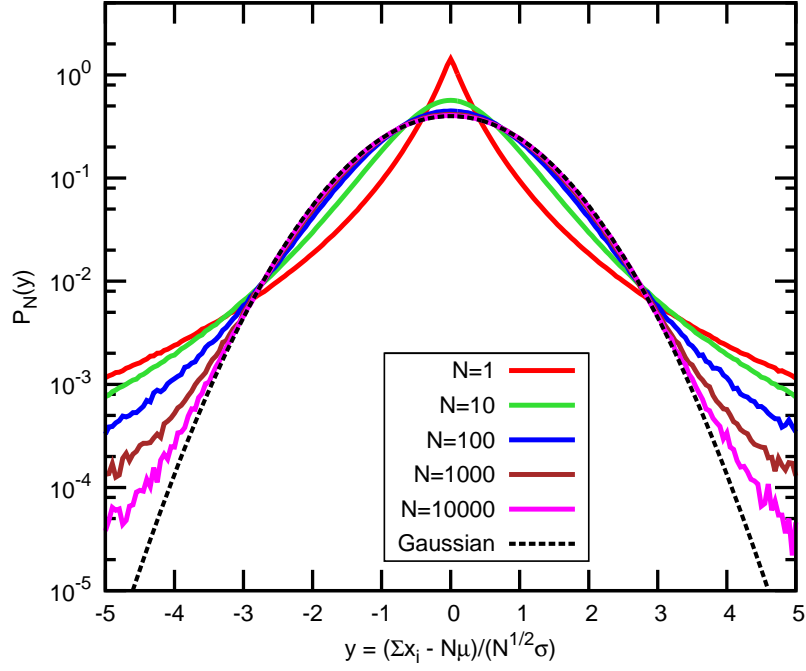


FIG. 6: Figure showing the approach to the central limit theorem for the distribution in Eq. (A15), which has mean, μ , equal to 0, and standard deviation, σ , equal to 1. The horizontal axis is the sum of N random variables divided by \sqrt{N} which, for all N , has zero mean and standard deviation unity. For large N the distribution approaches a Gaussian. However, convergence is non-uniform, and is extremely slow in the tails.

In practice, distributions that we meet in nature, have a much broader tail than that of the Gaussian distribution, which falls off very fast at large $|x - \mu|/\sigma$. As a result, even if the distribution of the sum approximates well a Gaussian in the central region for only modest values of N , it might take a much larger value of N to beat down the weight in the tail to the value of the Gaussian. Hence, even for moderate values of N , the probability of a deviation greater than σ can be significantly larger than that of the Gaussian distribution which is 32%. This caution will be important in Sec. III when we discuss the quality of fits.

We will illustrate the slow convergence of the distribution of the sum to a Gaussian in Fig. (6), in which the distribution of the individual variables x_i is

$$P(x) = \frac{3}{2} \frac{1}{(1 + |x|)^4}. \quad (\text{A15})$$

This has mean 0 and standard deviation 1, but moments higher than the second do not exist because the integrals diverge. For large N the distribution approaches a Gaussian, as expected, but convergence is very slow in the tails.

Appendix B: The number of degrees of freedom

Consider, for simplicity, a straight line fit, so we have to determine the values of a_0 and a_1 which minimize Eq. (73). The N terms in Eq. (73) are not statistically independent at the minimum because the values of a_0 and a_1 , given by Eq. (74), depend on the data points (x_i, y_i, σ_i) .

Consider the “residuals” defined by

$$\epsilon_i = \frac{y_i - a_0 - a_1 x_i}{\sigma}. \quad (\text{B1})$$

If the model were exact and we use the exact values of the parameters a_0 and a_1 the ϵ_i would be independent and each have a Gaussian distribution with zero mean and standard deviation unity.

However, choosing the *best-fit* values of a_0 and a_1 *from the data* according to Eq. (74) implies that

$$\sum_{i=1}^N \frac{1}{\sigma_i} \epsilon_i = 0, \quad (\text{B2a})$$

$$\sum_{i=1}^N \frac{x_i}{\sigma_i} \epsilon_i = 0, \quad (\text{B2b})$$

which are two *linear constraints* on the ϵ_i . This means that we only need to specify $N - 2$ of them to know them all. In the N dimensional space of the ϵ_i we have eliminated two directions, so there can be no Gaussian fluctuations along them. However the other $N - 2$ dimensions are unchanged, and will have the same Gaussian fluctuations as before. Thus χ^2 has the distribution of a sum of squares of $N - 2$ Gaussian random variables. We can intuitively understand why there are $N - 2$ degrees of freedom rather than N by considering the case of $N = 2$. The fit goes perfectly through the two points so one has $\chi^2 = 0$ exactly. This implies that there are zero degrees of freedom since, on average, each degree of freedom adds 1 to χ^2 .

Clearly this argument can be generalized to any fitting function which depends *linearly* on M fitting parameters, with the result that χ^2 has the distribution of a sum of squares of $N_{\text{DOF}} = N - M$ Gaussian random variables, in which the quantity N_{DOF} is called the “number of degrees of freedom”.

Even if the fitting function depends non-linearly on the parameters, this last result is often taken as a reasonable approximation.

Appendix C: The chi-squared distribution and the goodness of fit parameter Q

The χ^2 distribution for m degrees of freedom is the distribution of the sum of m independent random variables with a Gaussian distribution with zero mean and standard deviation unity. To determine this we write the distribution of the m variables x_i as

$$P(x_1, x_2, \dots, x_m) dx_1 dx_2 \dots dx_m = \frac{1}{(2\pi)^{m/2}} e^{-x_1^2/2} e^{-x_2^2/2} \dots e^{-x_m^2/2} dx_1 dx_2 \dots dx_m. \quad (\text{C1})$$

Converting to polar coordinates, and integrating over directions, we find the distribution of the radial variable to be

$$\tilde{P}(r) dr = \frac{S_m}{(2\pi)^{m/2}} r^{m-1} e^{-r^2/2} dr, \quad (\text{C2})$$

where S_m is the surface area of a unit m -dimensional sphere. To determine S_m we integrate Eq. (C2) over r , noting that $\tilde{P}(r)$ is normalized, which gives

$$S_m = \frac{2\pi^{m/2}}{\Gamma(m/2)}, \quad (\text{C3})$$

where $\Gamma(x)$ is the Euler gamma function defined by

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt. \quad (\text{C4})$$

From Eqs. (C2) and (C3) we have

$$\tilde{P}(r) = \frac{1}{2^{m/2-1}\Gamma(m/2)} r^{m-1} e^{-r^2/2}. \quad (\text{C5})$$

This is the distribution of r but we want the distribution of $\chi^2 \equiv \sum_i x_i^2 = r^2$. To avoid confusion of notation we write X for χ^2 , and define the χ^2 distribution for m variables as $P^{(m)}(X)$. We have $P^{(m)}(X) dX = \tilde{P}(r) dr$ so the χ^2 distribution for m degrees of freedom is

$$P^{(m)}(X) = \frac{\tilde{P}(r)}{dX/dr} = \frac{1}{2^{m/2}\Gamma(m/2)} X^{(m/2)-1} e^{-X/2} \quad (X > 0). \quad (\text{C6})$$

The χ^2 distribution is zero for $X < 0$. Using Eq. (C4) and the property of the gamma function that $\Gamma(n+1) = n\Gamma(n)$ one can show that

$$\int_0^\infty P^{(m)}(X) dX = 1, \quad (\text{C7a})$$

$$\langle X \rangle \equiv \int_0^\infty X P^{(m)}(X) dX = m, \quad (\text{C7b})$$

$$\langle X^2 \rangle \equiv \int_0^\infty X^2 P^{(m)}(X) dX = m^2 + 2m, \quad \text{so} \quad (\text{C7c})$$

$$\langle X^2 \rangle - \langle X \rangle^2 = 2m. \quad (\text{C7d})$$

From Eqs. (C7b) and (C7d) we see that typically χ^2 lies in the range $m - \sqrt{2m}$ to $m + \sqrt{2m}$. For large m the distribution approaches a Gaussian according to the central limit theory discussed in Appendix A. Typically one focuses on the value of χ^2 per degree freedom since this should be around unity independent of m .

The goodness of fit parameter is the probability that the specified value of χ^2 , or greater, could occur by random chance. From Eq. (C6) it is given by

$$Q = \frac{1}{2^{m/2}\Gamma(m/2)} \int_{\chi^2}^{\infty} X^{(m/2)-1} e^{-X/2} dX, \quad (\text{C8})$$

$$\boxed{= \frac{1}{\Gamma(m/2)} \int_{\chi^2/2}^{\infty} y^{(m/2)-1} e^{-y} dy,} \quad (\text{C9})$$

which is known as an incomplete gamma function. Code to generate the incomplete gamma function is given in Numerical Recipes [1]. There is also a built-in function to generate the goodness of fit parameter in the `scipy` package of `python` and in the graphics program `gnuplot`, see the scripts in Appendix G.

Note that $Q = 1$ for $\chi^2 = 0$ and $Q \rightarrow 0$ for $\chi^2 \rightarrow \infty$. Remember that m is the number of degrees of freedom, written as N_{DOF} elsewhere in these notes.

Appendix D: Asymptotic standard error and how to get correct error bars from gnuplot

Sometimes one does not have error bars on the data. Nonetheless, one can still use χ^2 fitting to get an *estimate* of those errors (assuming that they are all equal) and thereby also get an error bar on the fit parameters. The latter is called the “asymptotic standard error”. Assuming the same error bar σ_{ass} for all points, we determine σ_{ass} from the requirement that χ^2 per degree of freedom is precisely one, i.e. its mean value according to Eq. (C7b). This gives

$$1 = \frac{\chi^2}{N_{\text{DOF}}} = \frac{1}{N_{\text{DOF}}} \sum_{i=1}^N \left(\frac{y_i - f(x_i)}{\sigma_{\text{ass}}} \right)^2, \quad (\text{D1})$$

or, equivalently,

$$\boxed{\sigma_{\text{ass}}^2 = \frac{1}{N_{\text{DOF}}} \sum_{i=1}^N (y_i - f(x_i))^2.} \quad (\text{D2})$$

The error bars on the fit parameters are then obtained from Eq. (90), with the elements of U given by Eq. (76) in which σ_i is replaced by σ_{ass} . Equivalently, one can set the σ_i to unity in determining U from Eq. (76), and estimate the error on the fit parameters from

$$\boxed{\sigma_{\alpha}^2 = (U)_{\alpha\alpha}^{-1} \sigma_{\text{ass}}^2,} \quad (\text{asymptotic standard error}). \quad (\text{D3})$$

A simple example of the use of the asymptotic standard error in a situation where we don't know the error on the data points, is fitting to a constant, i.e. *determining the average of a set of data*, which we already discussed in detail in Sec. II. In this case we have

$$U_{00} = N, \quad v_0 = \sum_{i=1}^N y_i, \quad (\text{D4})$$

so the only fit parameter is

$$a_0 = \frac{v_0}{U_{00}} = \frac{1}{N} \sum_{i=1}^N y_i = \bar{y}, \quad (\text{D5})$$

which gives, naturally enough, the average of the data points, \bar{y} . The number of degrees of freedom is $N - 1$, since there is one fit parameter, so

$$\sigma_{\text{ass}}^2 = \frac{1}{N-1} \sum_{i=1}^N (y_i - \bar{y})^2, \quad (\text{D6})$$

and hence the square of the error on a_0 is given, from Eq. (D3), by

$$\sigma_0^2 = \frac{1}{U_{00}} \sigma_{\text{ass}}^2 = \frac{1}{N(N-1)} \sum_{i=1}^N (y_i - \bar{y})^2, \quad (\text{D7})$$

which is precisely the expression for the error in the mean of a set of data given in Eq. (16).

I now mention that a popular plotting program, **gnuplot**, which also does fits, presents error bars on the fit parameters incorrectly if there are error bars on the data. Whether or not there are error bars on the points, **gnuplot** presents the “asymptotic standard error” on the fit parameters. **Gnuplot** calculates the elements of U correctly from Eq. (76) including the error bars, but then apparently also determines an “assumed error” from an expression like Eq. (D2) but including the error bars, i.e.

$$\sigma_{\text{ass}}^2 = \frac{1}{N_{\text{DOF}}} \sum_{i=1}^N \left(\frac{y_i - f(x_i)}{\sigma_i} \right)^2 = \frac{\chi^2}{N_{\text{DOF}}}, \quad (\text{gnuplot}). \quad (\text{D8})$$

Hence **gnuplot**'s σ_{ass}^2 is just the chi-squared per degree of freedom. The error bar (squared) quoted by **gnuplot** is $(U)_{\alpha\alpha}^{-1} \sigma_{\text{ass}}^2$, as in Eq. (D3). However, this is wrong since the error bars on the data points have *already* been included in calculating the elements of U , so the error on the fit parameter α should be $(U)_{\alpha\alpha}^{-1}$. Hence,

to get correct error bars on fit parameters from **gnuplot** when there are error bars on the points, you have to divide **gnuplot**'s asymptotic standard errors by the square root of the chi-squared per degree of freedom (which **gnuplot** calls `FIT_STDFIT` and, fortunately, computes correctly).

I have checked this statement by comparing with results for Numerical Recipes routines, and also, for straight-line fits, by my own implementation of the formulae. It is curious that I found no hits on this topic when googling the internet. Can no one else have come across this problem? Correction of `gnuplot` error bars is implemented in the `gnuplot` scripts in Appendix G

The need to correct `gnuplot`'s error bars applies to linear as well as non-linear models.

I recently learned that error bars on fit parameters given by the routine `curve_fit` of `python` also have to be corrected in the same way. This is shown in two of the `python` scripts in appendix G. Curiously, a different `python` fitting routine, `leastsq`, gives the error bars correctly.

Appendix E: The distribution of fitted parameters determined from simulated datasets

In this section we derive the equation for the distribution of fitted parameters determined from simulated datasets, Eq. (100), assuming an arbitrary linear model, see Eq. (96). Projecting on to a single fitting parameter, as above, this corresponds to the lower figure in Fig. 2.

We have *one* set of y -values, $y_i^{(0)}$, for which the fit parameters are $\vec{a}^{(0)}$. We then generate an *ensemble* of simulated data sets, y_i^S , assuming the data has Gaussian noise with standard deviation σ_i centered on the actual data values $y_i^{(0)}$. We ask for the probability that the fit to one of the simulated data sets has parameters \vec{a}^S .

This probability distribution is given by

$$P(\vec{a}^S) = \prod_{i=1}^N \left\{ \frac{1}{\sqrt{2\pi}\sigma_i} \int_{-\infty}^{\infty} dy_i^S \exp \left[-\frac{(y_i^S - y_i^{(0)})^2}{2\sigma_i^2} \right] \right\} \prod_{\alpha=1}^M \delta \left(\sum_{\beta} U_{\alpha\beta} a_{\beta}^S - v_{\alpha}^S \right) \det U, \quad (\text{E1})$$

where the factor in curly brackets is (an integral over) the probability distribution of the data points y_i^S , and the delta functions project out those sets of data points which have a particular set of fitted parameters, see Eq. (99). The factor of $\det U$ is a Jacobian to normalize the distribution. Using the integral representation of the delta function, and writing explicitly the expression for v_{α} from Eq. (98), one has

$$P(\vec{a}^S) = \prod_{i=1}^N \left\{ \frac{1}{\sqrt{2\pi}\sigma_i} \int_{-\infty}^{\infty} dy_i^S \exp \left[-\frac{(y_i^S - y_i^{(0)})^2}{2\sigma_i^2} \right] \right\} \times \quad (\text{E2})$$

$$\prod_{\alpha=1}^M \left(\frac{1}{2\pi} \int_{-\infty}^{\infty} dk_{\alpha} \exp \left[ik_{\alpha} \left(\sum_{\beta} U_{\alpha\beta} a_{\beta}^S - \sum_{i=1}^N \frac{y_i^S X_{\alpha}(x_i)}{\sigma_i^2} \right) \right] \right) \det U. \quad (\text{E3})$$

We carry out the y integrals by “completing the square”,

$$P(\vec{a}^S) = \prod_{\alpha=1}^M \left(\frac{1}{2\pi} \int_{-\infty}^{\infty} dk_{\alpha} \right) \prod_{i=1}^N \left\{ \frac{1}{\sqrt{2\pi}\sigma_i} \int_{-\infty}^{\infty} dy_i^S \exp \left[-\frac{\left(y_i^S - y_i^{(0)} + i\vec{k} \cdot \vec{X}(x_i) \right)^2}{2\sigma_i^2} \right] \right\} \times \quad (\text{E4})$$

$$\exp \left[-\frac{1}{2\sigma_i^2} \left(\left(\vec{k} \cdot \vec{X}(i) \right)^2 + 2i \left(\vec{k} \cdot \vec{X}(x_i) \right) y_i^{(0)} \right) \right] \times \exp \left[i \sum_{\alpha,\beta} k_{\alpha} U_{\alpha\beta} a_{\beta}^S \right] \det U. \quad (\text{E5})$$

Doing the y^S -integrals, the factors in curly brackets are equal to unity. Using Eqs. (97) and (98) and the fact that the $U_{\alpha\beta}$ are independent of the y_i^S , we then get

$$P(\vec{a}^S) = \prod_{\alpha=1}^M \left(\frac{1}{2\pi} \int_{-\infty}^{\infty} dk_{\alpha} \right) \exp \left[-\frac{1}{2} \sum_{\alpha,\beta} k_{\alpha} U_{\alpha\beta} k_{\beta} + i \sum_{\alpha,\beta} k_{\alpha} \delta v_{\beta}^S \right] \det U, \quad (\text{E6})$$

where

$$\delta v_{\beta}^S \equiv v_{\beta}^S - v_{\beta}^{(0)}, \quad (\text{E7})$$

with

$$v_{\alpha}^{(0)} = \sum_{i=1}^N \frac{y_i^{(0)} X_{\alpha}(x_i)}{\sigma_i^2}. \quad (\text{E8})$$

We do the k -integrals by working in the basis in which U is diagonal. The result is

$$P(\vec{a}^S) = \frac{(\det U)^{1/2}}{(2\pi)^{m/2}} \exp \left[-\frac{1}{2} \sum_{\alpha,\beta} \delta v_{\alpha}^S (U^{-1})_{\alpha\beta} \delta v_{\beta}^S \right]. \quad (\text{E9})$$

Using Eq. (99) and the fact that U is symmetric we get our final result

$$\boxed{P(\vec{a}^S) = \frac{(\det U)^{1/2}}{(2\pi)^{m/2}} \exp \left[-\frac{1}{2} \sum_{\alpha,\beta} \delta a_{\alpha}^S U_{\alpha\beta} \delta a_{\beta}^S \right]}, \quad (\text{E10})$$

which is Eq. (100), including the normalization constant in front of the exponential.

Appendix F: The distribution of fitted parameters from repeated sets of measurements

In this section we derive the equation for the distribution of fitted parameters determined in the hypothetical situation that one has many actual data sets. Projecting on to a single fitted parameter, this corresponds to the upper figure in Fig. 2.

The exact value of the data is $y_i^{\text{true}} = \vec{a}^{\text{true}} \cdot \vec{X}(x_i)$, and the distribution of the y_i in an actual data set, which differs from y_i^{true} because of noise, has a distribution, assumed Gaussian here,

centered on y_i^{true} with standard deviation σ_i . Fitting each of these real data sets, the probability distribution for the fitted parameters is given by

$$P(\vec{a}) = \prod_{i=1}^N \left\{ \frac{1}{\sqrt{2\pi}\sigma_i} \int_{-\infty}^{\infty} dy_i \exp \left[-\frac{(y_i - \vec{a}^{\text{true}} \cdot \vec{X}(x_i))^2}{2\sigma_i^2} \right] \right\} \prod_{\alpha=1}^M \delta \left(\sum_{\beta} U_{\alpha\beta} a_{\beta} - v_{\alpha} \right) \det U, \quad (\text{F1})$$

see Eq. (E1) for an explanation of the various factors. Proceeding as in Appendix E we have

$$P(\vec{a}) = \prod_{i=1}^N \left\{ \frac{1}{\sqrt{2\pi}\sigma_i} \int_{-\infty}^{\infty} dy_i \exp \left[-\frac{(y_i - \vec{a}^{\text{true}} \cdot \vec{X}(x_i))^2}{2\sigma_i^2} \right] \right\} \times \quad (\text{F2})$$

$$\prod_{\alpha=1}^M \left(\frac{1}{2\pi} \int_{-\infty}^{\infty} dk_{\alpha} \exp \left[ik_{\alpha} \left(\sum_{\beta} U_{\alpha\beta} a_{\beta} - \sum_{i=1}^N \frac{y_i X_{\alpha}(x_i)}{\sigma_i^2} \right) \right] \right) \det U, \quad (\text{F3})$$

and doing the y -integrals by completing the square gives

$$P(\vec{a}) = \prod_{\alpha=1}^M \left(\frac{1}{2\pi} \int_{-\infty}^{\infty} dk_{\alpha} \right) \times \quad (\text{F4})$$

$$\exp \left[-\frac{1}{2\sigma_i^2} \left((\vec{k} \cdot \vec{X}(i))^2 + 2i (\vec{k} \cdot \vec{X}(x_i)) (\vec{a}^{\text{true}} \cdot \vec{X}(x_i)) \right) \right] \times \exp \left[i \sum_{\alpha,\beta} k_{\alpha} U_{\alpha\beta} a_{\beta} \right] \det U. \quad (\text{F5})$$

Using Eq. (97) we then get

$$P(\vec{a}) = \prod_{\alpha=1}^M \left(\frac{1}{2\pi} \int_{-\infty}^{\infty} dk_{\alpha} \right) \exp \left[-\frac{1}{2} \sum_{\alpha,\beta} k_{\alpha} U_{\alpha\beta} k_{\beta} + i \sum_{\alpha,\beta} k_{\alpha} U_{\alpha\beta} \delta a_{\beta} \right] \det U, \quad (\text{F6})$$

where

$$\delta a_{\beta} \equiv a_{\beta} - a_{\beta}^{\text{true}}. \quad (\text{F7})$$

The k -integrals are done by working in the basis in which U is diagonal. The result is

$$\boxed{P(\vec{a}) = \frac{(\det U)^{1/2}}{(2\pi)^{m/2}} \exp \left[-\frac{1}{2} \sum_{\alpha,\beta} \delta a_{\alpha} U_{\alpha\beta} \delta a_{\beta} \right]}. \quad (\text{F8})$$

In other words, the distribution of the fitted parameters obtained from many sets of actual data, about the *true* value \vec{a}^{true} is a Gaussian. Since we are assuming a linear model, the matrix of coefficients $U_{\alpha\beta}$ is a constant, and so the distribution in Eq. (F8) is the *same* as in Eq. (E10).

Hence

For a linear model with Gaussian noise, the distribution of fitted parameters, obtained from simulated data sets, relative to *value from the one actual data set*, is the same as the distribution of parameters from many actual data sets relative to *the true value*, see Fig. 2.

This result is also valid for a non-linear model if the range of parameter values needed is sufficiently small that the model can be represented by an effective one. It is usually assumed to be a reasonable approximation even if this condition is not fulfilled.

Appendix G: Scripts for some data analysis and fitting tasks

In this appendix I give sample scripts using perl, python and gnuplot for some basic data analysis and fitting tasks. I include output from the scripts when acting on certain datasets which are available on the web.

Note “`this_file_name`” refers to the name of the script being displayed (whatever you choose to call it.)

1. Scripts for a jackknife analysis

The script reads in values of x on successive lines of the input file and computes $\langle x^4 \rangle / \langle x^2 \rangle^2$, including an error bar computed using the jackknife method.

a. Perl

```
#!/usr/bin/perl
#
# Usage: "this_file_name data_file"
# (make the script executable; otherwise you have to preface the command with "perl")
#
$n = 0;
$x2_tot = 0; $x4_tot = 0;
#
# read in the data
#
while(<>) # Note this very convenient perl command which reads each line of
        # of each input file in the command line
{
    @line = split;
    $x2[$n] = $line[0]**2;
    $x4[$n] = $x2[$n]**2;
    $x2_tot += $x2[$n];
    $x4_tot += $x4[$n];
```

```

    $n++;
}
#
# Do the jackknife estimates
#
for ($i = 0; $i < $n; $i++)
{
    $x2_jack[$i] = ($x2_tot - $x2[$i]) / ($n - 1);
    $x4_jack[$i] = ($x4_tot - $x4[$i]) / ($n - 1);
}
$x2_av = $x2_tot / $n;    # Do the overall averages
$x4_av = $x4_tot / $n;
$g_av = $x4_av / $x2_av**2;

$g_jack_av = 0; $g_jack_err = 0; # Do the final jackknife estimate
for ($i = 0; $i < $n; $i++)
{
    $dg = $x4_jack[$i] / $x2_jack[$i]**2;
    $g_jack_av += $dg;
    $g_jack_err += $dg**2;
}
$g_jack_av /= $n;
$g_jack_err /= $n;
$g_jack_err = sqrt(($n - 1) * abs($g_jack_err - $g_jack_av**2));

printf " Overall average is    %8.4f\n", $g_av;
printf " Jackknife average is %8.4f +/- %6.4f \n", $g_jack_av, $g_jack_err;

```

Executing this file on the data in <http://physics.ucsc.edu/~peter/bad-honnef/data.HW2> gives

```

Overall average is    1.8215
Jackknife average is  1.8215 +/- 0.0368

```

b. Python

```

#
# Program written by Matt Wittmann
#
# Usage: "python this_file_name data_file"
#
import fileinput
from math import *

x2 = []; x2_tot = 0.
x4 = []; x4_tot = 0.
for line in fileinput.input(): # read in each line in each input file.
    # similar to perl's while(<>)
    line = line.split()
    x2_i = float(line[0])**2
    x4_i = x2_i**2

```

```

        x2.append(x2_i)          # put x2_i as the i-th element in an array x2
        x4.append(x4_i)
        x2_tot += x2_i
        x4_tot += x4_i
n = len(x2)                      # the number of lines read in
#
# Do the jackknife estimates
#
x2_jack = []
x4_jack = []
for i in xrange(n):
    x2_jack.append((x2_tot - x2[i]) / (n - 1))
    x4_jack.append((x4_tot - x4[i]) / (n - 1))

x2_av = x2_tot / n              # do the overall averages
x4_av = x4_tot / n
g_av = x4_av / x2_av**2
g_jack_av = 0.; g_jack_err = 0.

for i in xrange(n):            # do the final jackknife averages
    dg = x4_jack[i] / x2_jack[i]**2
    g_jack_av += dg
    g_jack_err += dg**2

g_jack_av /= n
g_jack_err /= n
g_jack_err = sqrt((n - 1) * abs(g_jack_err - g_jack_av**2))

print " Overall average is    %8.4f" % g_av
print " Jackknife average is %8.4f +/- %6.4f" % (g_jack_av, g_jack_err)

```

The output is the same as for the perl script.

2. Scripts for a straight-line fit

a. Perl, writing out the formulae by hand

```

#!/usr/bin/perl
#
# Usage: "this_file_name data_file"
# (make the script executable; otherwise preface the command with "perl")
#
# Does a straight line fit to data in "data_file" each line of which contains
# data for one point, x_i, y_i, sigma_i
#
$n = 0;
while(<>)    # read in the lines of data
{
    @line = split; # split the line to get x_i, y_i, sigma_i

```



```

    $x[$n] = $line[0];
    $y[$n] = $line[1];
    $err[$n] = $line[2];
    $err2 = $err[$n]**2;    # compute the necessary sums over the data
    $s += 1 / $err2;
    $sumx += $x[$n] / $err2 ;
    $sumy += $y[$n] / $err2 ;
    $sumxx += $x[$n]*$x[$n] / $err2 ;
    $sumxy += $x[$n]*$y[$n] / $err2 ;
    $n++;
}

$delta = $s * $sumxx - $sumx * $sumx ; # compute the slope and intercept
$c = ($sumy * $sumxx - $sumx * $sumy) / $delta ;
$m = ($s * $sumxy - $sumx * $sumy) / $delta ;
$errm = sqrt($s / $delta) ;
$errc = sqrt($sumxx / $delta) ;

printf ("slope      = %10.4f +/- %7.4f \n", $m, $errm); # print the results
printf ("intercept = %10.4f +/- %7.4f \n\n", $c, $errc);

$NDF = $n - 2; # the no. of degrees of freedom is n - no. of fit params
$chisq = 0;    # compute the chi-squared
for ($i = 0; $i < $n; $i++)
{
    $chisq += (($y[$i] - $m*$x[$i] - $c)/$err[$i])**2;
}
$chisq /= $NDF;
printf ("chi squared / NDF = %7.4lf \n", $chisq);

```

Acting with this script on the data in <http://physics.ucsc.edu/~peter/bad-honnef/data.HW3> gives

```

slope      =      5.0022 +/-  0.0024
intercept =      0.9046 +/-  0.2839

```

```

chi squared / NDF =  1.0400

```

b. Python, writing out the formulae by hand

```

#
# Program written by Matt Wittmann
#
# Usage: "python this_file_name data_file"
#
# Does a straight-line fit to data in "data_file", each line of which contains
# the data for one point, x_i, y_i, sigma_i
#
import fileinput

```

```

from math import *

x = []
y = []
err = []
s = sumx = sumy = sumxx = sumxy = 0.

for line in fileinput.input():    # read in the data, one line at a time
    line = line.split()          # split the line
    x_i = float(line[0]);    x.append(x_i)
    y_i = float(line[1]);    y.append(y_i)
    err_i = float(line[2]); err.append(err_i)
    err2 = err_i**2
    s += 1 / err2              # do the necessary sums over data points
    sumx  += x_i / err2
    sumy  += y_i / err2
    sumxx += x_i*x_i / err2
    sumxy += x_i*y_i / err2

n = len(x)                    # n is the number of data points
delta = s * sumxx - sumx * sumx # compute the slope and intercept
c = (sumy * sumxx - sumx * sumy) / delta
m = (s * sumxy - sumx * sumy) / delta
errm = sqrt(s / delta)
errc = sqrt(sumxx / delta)

print "slope      = %10.4f +/- %7.4f " % (m, errm)
print "intercept = %10.4f +/- %7.4f \n" % (c, errc)

NDF = n - 2                  # the number of degrees of freedom is n - 2
chisq = 0.

for i in xrange(n):          # compute chi-squared
    chisq += ((y[i] - m*x[i] - c)/err[i])**2;

chisq /= NDF
print "chi squared / NDF = %7.4lf " % chisq

```

The results are identical to those from the perl script.

c. Python, using a built-in routine from scipy

```

#
# Python program written by Matt Wittmann
#
# Usage: "python this_file_name data_file"
#
# Does a straight-line fit to data in "data_file", each line of which contains
# the data for one point, x_i, y_i, sigma_i.

```

```

#
# Uses the built-in routine "curve_fit" in the scipy package. Note that this
# requires the error bars to be corrected, as with gnuplot
#
from pylab import *
from scipy.optimize import curve_fit

fname = sys.argv[1] if len(sys.argv) > 1 else 'data.txt'
x, y, err = np.loadtxt(fname, unpack=True) # read in the data
n = len(x)

p0 = [5., 0.1] # initial values of parameters
f = lambda x, c, m: c + m*x                # define the function to be fitted
                                           # note python's lambda notation
p, covm = curve_fit(f, x, y, p0, err)      # do the fit
c, m = p
chisq = sum(((f(x, c, m) - y)/err)**2)     # compute the chi-squared
chisq /= n - 2                             # divide by no. of DOF
errc, errm = sqrt(diag(covm)/chisq)        # correct the error bars

print "slope      = %10.4f +/- %7.4f " % (m, errm)
print "intercept = %10.4f +/- %7.4f \n" % (c, errc)
print "chi squared / NDF = %7.4lf " % chisq

```

The results are identical to those from the above scripts.

d. Gnuplot

```

#
# Gnuplot script to plot points, do a straight-line fit, and display the
# points, fit, fit parameters, error bars, chi-squared per degree of freedom,
# and goodness of fit parameter on the plot.
#
# Usage: "gnuplot this_file_name"
#
# The data is assumed to be a file "data.HW3", each line containing
# information for one point (x_i, y_i, sigma_i). The script produces a
# postscript file, called here "HW3b.eps".
#
set size 1.0, 0.6
set terminal postscript portrait enhanced font 'Helvetica,16'
set output "HW3b.eps"
set fit errorvariables # needed to be able to print error bars
f(x) = a + b * x      # the fitting function
fit f(x) "data.HW3" using 1:2:3 via a, b # do the fit
set xlabel "x"
set ylabel "y"
ndf = FIT_NDF          # Number of degrees of freedom
chisq = FIT_STDFIT**2 * ndf # chi-squared

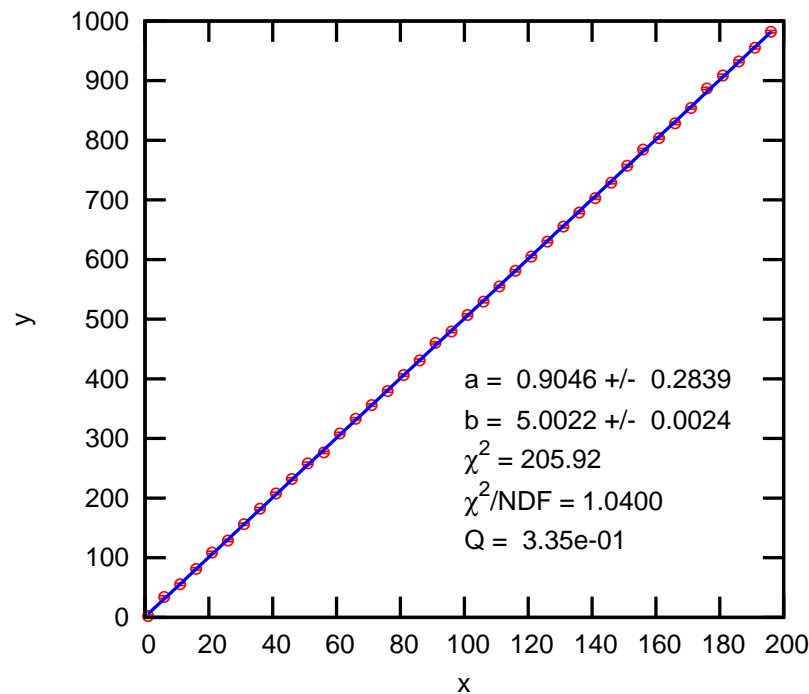
```

```

Q = 1 - igamma(0.5 * ndf, 0.5 * chisq) # the quality of fit parameter Q
#
# Below note how the error bars are (a) corrected by dividing by
# FIT_STDFIT, and (b) are displayed on the plot, in addition to the fit
# parameters, neatly formatted using sprintf.
#
set label sprintf("a = %7.4f +/- %7.4f", a, a_err/FIT_STDFIT) at 100, 400
set label sprintf("b = %7.4f +/- %7.4f", b, b_err/FIT_STDFIT) at 100, 330
set label sprintf("{/Symbol c}^2 = %6.2f", chisq) at 100, 270
set label sprintf("{/Symbol c}^2/NDF = %6.4f", FIT_STDFIT**2) at 100, 200
set label sprintf("Q = %9.2e", Q) at 100, 130
plot \
          # Plot the data and fit
"data.HW3" using 1:2:3 every 5 with errorbars notitle pt 6 lc rgb "red" lw 2, \
f(x) notitle lc rgb "blue" lw 4 lt 1

```

The plot below shows the result of acting with this gnuplot script on a the data in <http://physics.ucsc.edu/~peter/bad-honnef/data.HW3>. The results agree with those of the



other scripts.

3. Scripts for a fit to a non-linear model

We read in lines of data each of which contains three entries x_i, y_i and σ_i . These are fitted to the form

$$y = T_c + A/x^\omega, \quad (\text{G1})$$

to determine the best values of T_c, A and ω .

a. *Python*

```

#
# Python program written by Matt Wittmann
#
# Usage: "python this_file_name data_file"
#
# Does a fit to the non-linear model
#
#  $y = T_c + A / x^{**w}$ 
#
# to the data in "data_file", each line of which contains the data for one point,
# x_i, y_i, sigma_i.
#
# Uses the built-in routine "curve_fit" in the scipy package. Note that this
# requires the error bars to be corrected, as with gnuplot
#
from pylab import *
from scipy.optimize import curve_fit
from scipy.stats import chi2

fname = sys.argv[1] if len(sys.argv) > 1 else 'data.txt'
x, y, err = np.loadtxt(fname, unpack=True) # read in the data
n = len(x)                                # the number of data points

p0 = [-0.25, 0.2, 2.8]                    # initial values of parameters
f = lambda x, Tc, w, A: Tc + A/x**w       # define the function to be fitted
                                           # note python's lambda notation
p, covm = curve_fit(f, x, y, p0, err)     # do the fit
Tc, w, A = p
chisq = sum(((f(x, Tc, w, A) - y)/err)**2) # compute the chi-squared
ndf = n - len(p)                          # no. of degrees of freedom
Q = 1. - chi2.cdf(chisq, ndf)              # compute the quality of fit parameter Q
chisq = chisq / ndf                       # compute chi-squared per DOF
Tcerr, werr, Aerr = sqrt(diag(covm)/chisq) # correct the error bars

print 'Tc = %10.4f +/- %7.4f' % (Tc, Tcerr)
print 'A = %10.4f +/- %7.4f' % (A, Aerr)
print 'w = %10.4f +/- %7.4f' % (w, werr)
print 'chi squared / NDF = %7.4lf' % chisq
print 'Q = %10.4f' % Q

```

When applied to the data in <http://physics.ucsc.edu/~peter/bad-honnef/data.HW4> the output is

```

Tc =      -0.2570 +/-   1.4775
A  =       2.7878 +/-   0.8250
w  =       0.2060 +/-   0.3508
chi squared / NDF =   0.2541
Q   =       0.9073

```

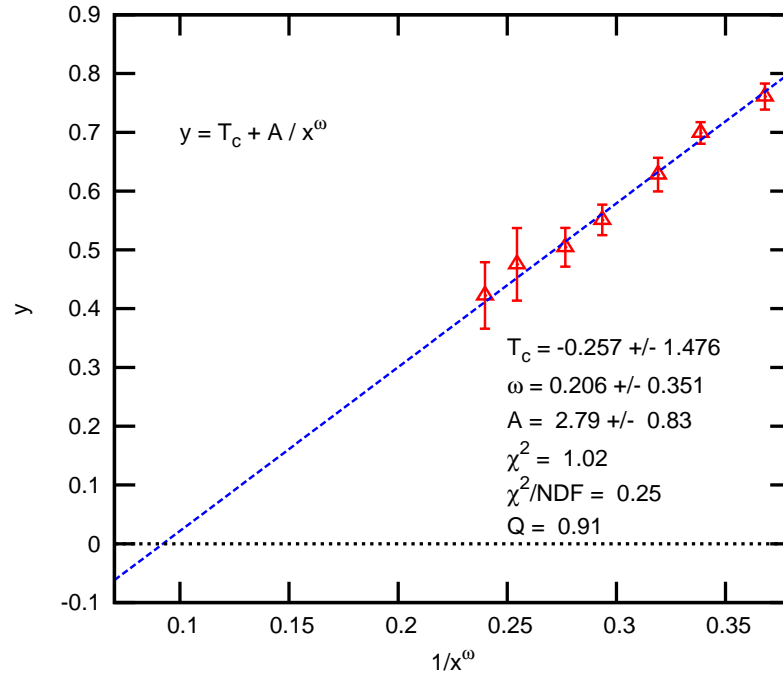
b. Gnuplot

```

#
# Gnuplot script to plot points, do a fit to a non-linear model
#
#  $y = T_c + A / x^w$ 
#
# with respect to  $T_c$ ,  $A$  and  $w$ , and display the points, fit, fit parameters,
# error bars, chi-squared per degree of freedom, and goodness of fit parameter
# on the plot.
#
# Here the data is assumed to be a file "data.HW4", each line containing
# information for one point ( $x_i$ ,  $y_i$ ,  $\sigma_i$ ). The script produces a
# postscript file, called here "HW4a.eps".
#
set size 1.0, 0.6
set terminal postscript portrait enhanced
set output "HW4a.eps"
set fit errorvariables      # needed to be able to print error bars
f(x) =  $T_c + A / x^w$       # the fitting function
set xlabel "1/ $x^{\text{/Symbol w}}$ "
set ylabel "y"
set label "y =  $T_c + A / x^{\text{/Symbol w}}$ " at 0.1, 0.7
Tc = 0.3                    # need to specify initial values
A = 1
w = 0.2
fit f(x) "data.HW4" using 1:2:3 via Tc, A, w # do the fit
set xrange [0.07:0.38]
g(x) =  $T_c + A * x$ 
h(x) =  $0 + 0 * x$ 
ndf = FIT_NDF               # Number of degrees of freedom
chisq = FIT_STDFIT**2 * ndf # chi-squared
Q = 1 - igamma(0.5 * ndf, 0.5 * chisq) # the quality of fit parameter Q
#
# Below note how the error bars are (a) corrected by dividing by
# FIT_STDFIT, and (b) are displayed on the plot, in addition to the fit
# parameters, neatly formatted using sprintf.
#
set label sprintf("T_c = %5.3f +/- %5.3f",Tc, Tc_err/FIT_STDFIT) at 0.25, 0.33
set label sprintf("{/Symbol w} = %5.3f +/- %5.3f",w, w_err/FIT_STDFIT) at 0.25, 0.27
set label sprintf("A = %5.2f +/- %5.2f",A, A_err/FIT_STDFIT) at 0.25, 0.21
set label sprintf("{/Symbol c}^2 = %5.2f", chisq) at 0.25, 0.15
set label sprintf("{/Symbol c}^2/NDF = %5.2f", FIT_STDFIT**2) at 0.25, 0.09
set label sprintf("Q = %5.2f", Q) at 0.25, 0.03
#
# Plot the data and the fit
#
plot "data.HW4" using (1/$1**w):2:3 with errorbars notitle lc rgb "red" lw 3 pt 8 ps 1.5, \
g(x) notitle lc rgb "blue" lw 3 lt 2 , \
h(x) notitle lt 3 lw 4

```

The plot below shows the result of acting with this gnuplot script on the data at <http://physics.ucsc.edu/~peter/bad-honnef/data.HW4>. The results agree with those of the



python script above.

Acknowledgments

I'm grateful to Alexander Hartmann for inviting me to give a lecture at the Bad Honnef School on "Efficient Algorithms in Computational Physics", which provided the motivation to write up these notes, and also for a helpful comment on the need to resample the data to get error bars from fits to non-linear models. I would also like to thank Matt Wittmann for helpful discussions about fitting and data analysis using `python` and for permission to include his python codes. I am also grateful to Wittmann and Christoph Norrenbrock for helpful comments on an earlier version of the manuscript.

-
- [1] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C, 2nd Ed.* (Cambridge University Press, Cambridge, 1992).
 - [2] V. Ambegaokar and M. Troyer, *Estimating errors reliably in Monte Carlo simulations of the Ehrenfest model*, Am. J. Phys. **78**, 150 (2009).